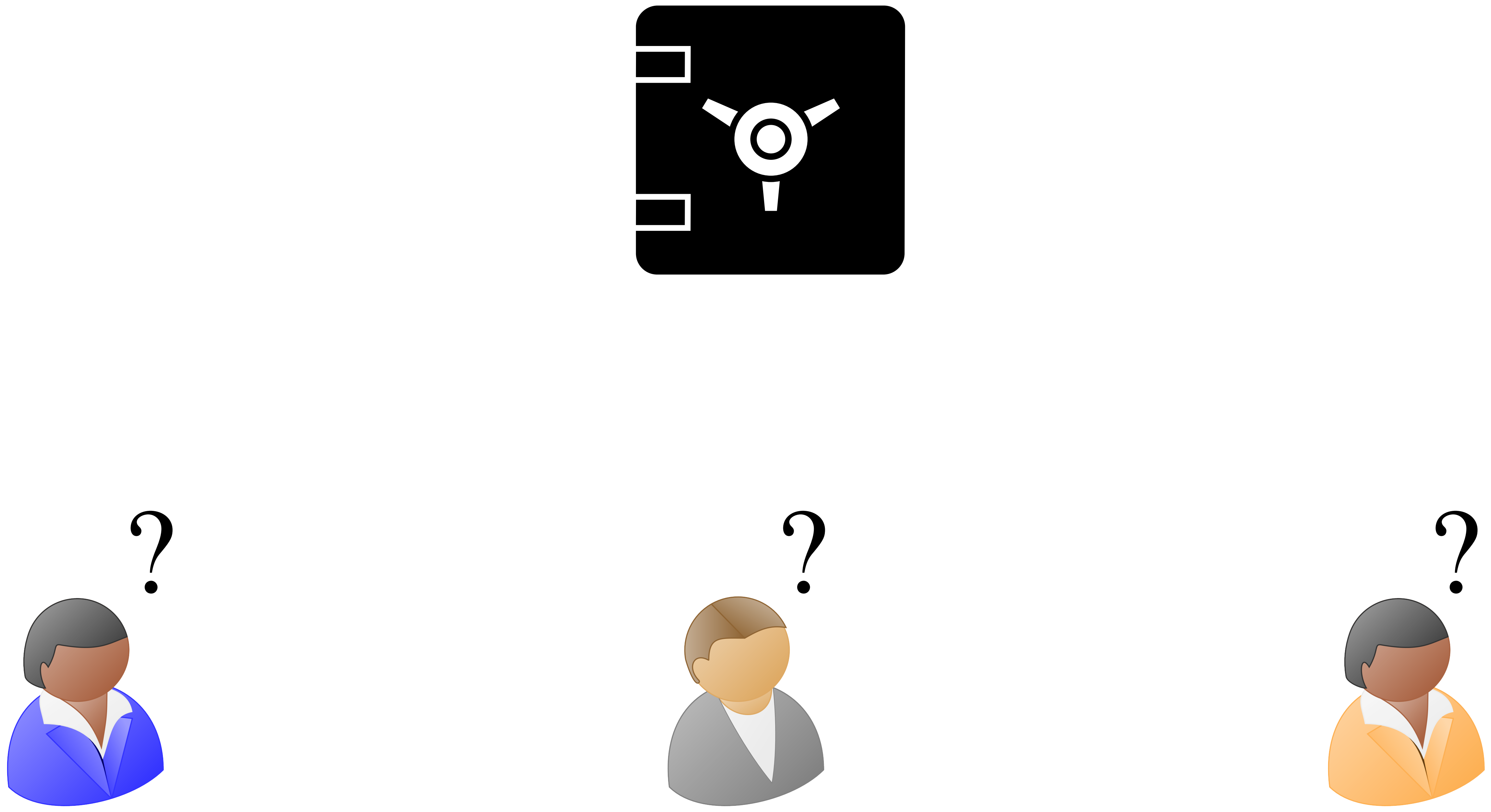
# Practical Cryptographic Systems

## Secret Sharing and Multi-party Computation

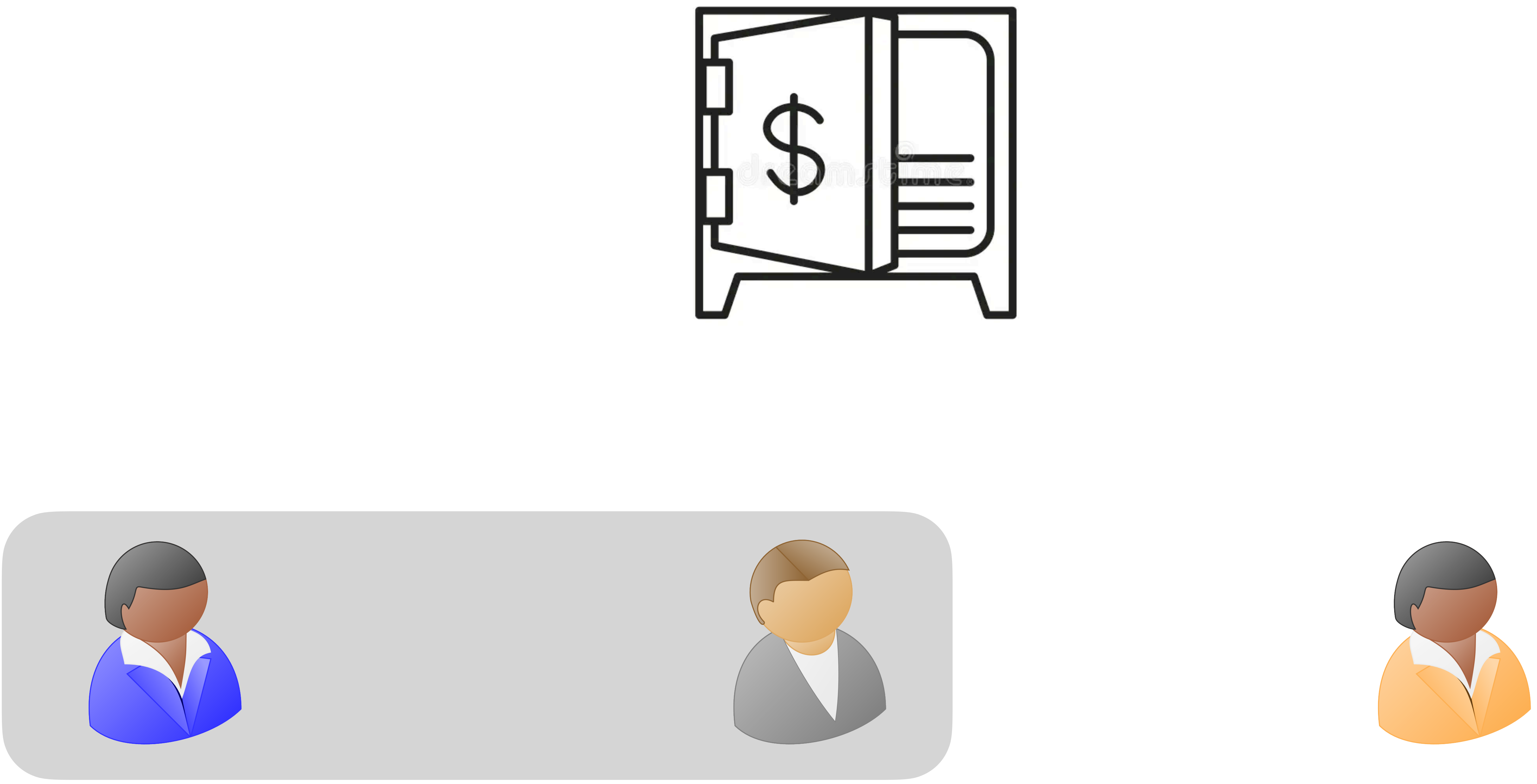
Instructor: Matthew Green

# Secret Sharing: Motivation




No single manager should be able to open the safe by themselves

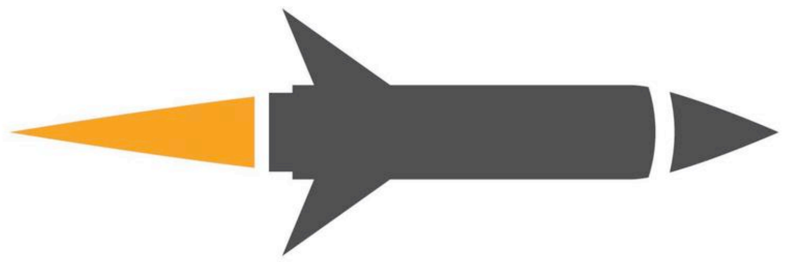
# Secret Sharing: Motivation



No single manager should be able to open the safe by themselves

But any two of them should be be able to open the safe

# Secret Sharing: Motivation

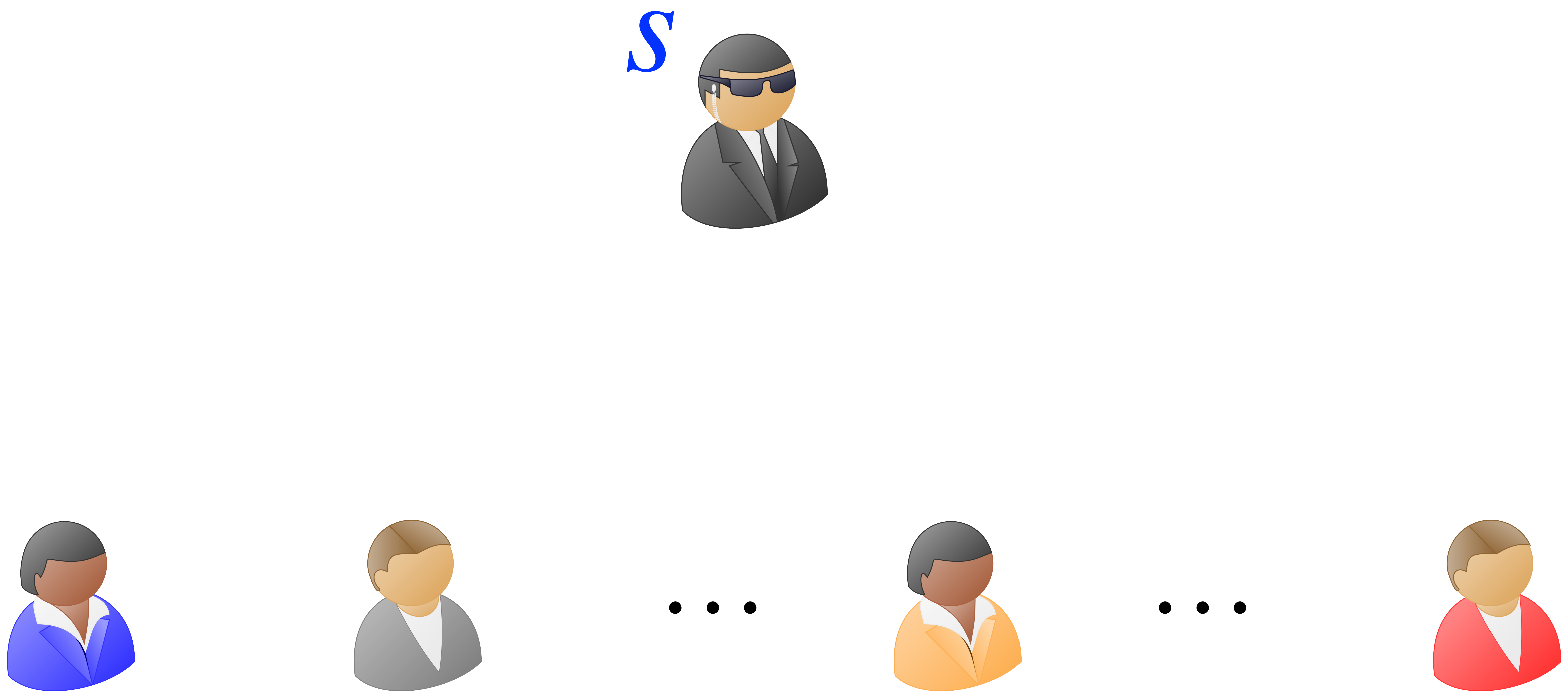


President

Secretary of Defense

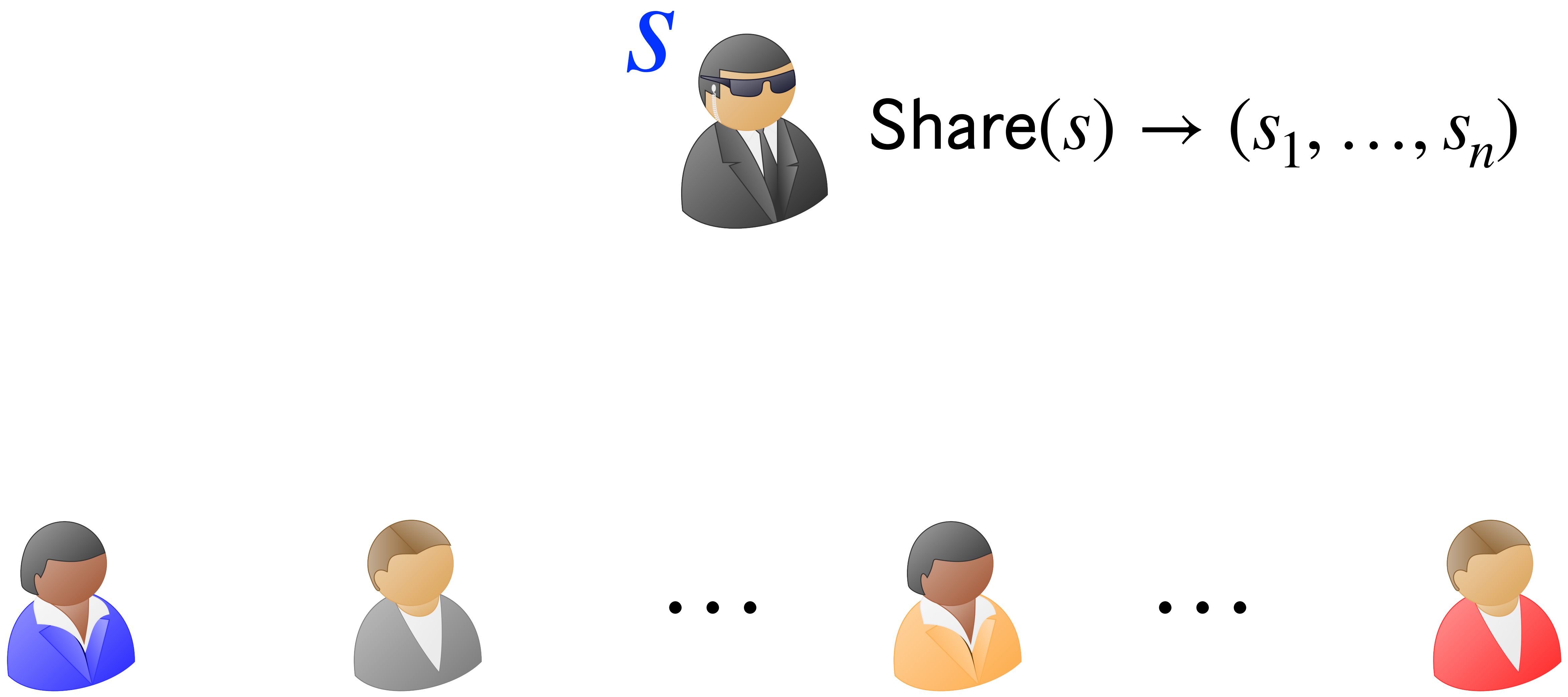
Head of Joint Chief of Staff

# $k$-out-of-$n$ Secret Sharing



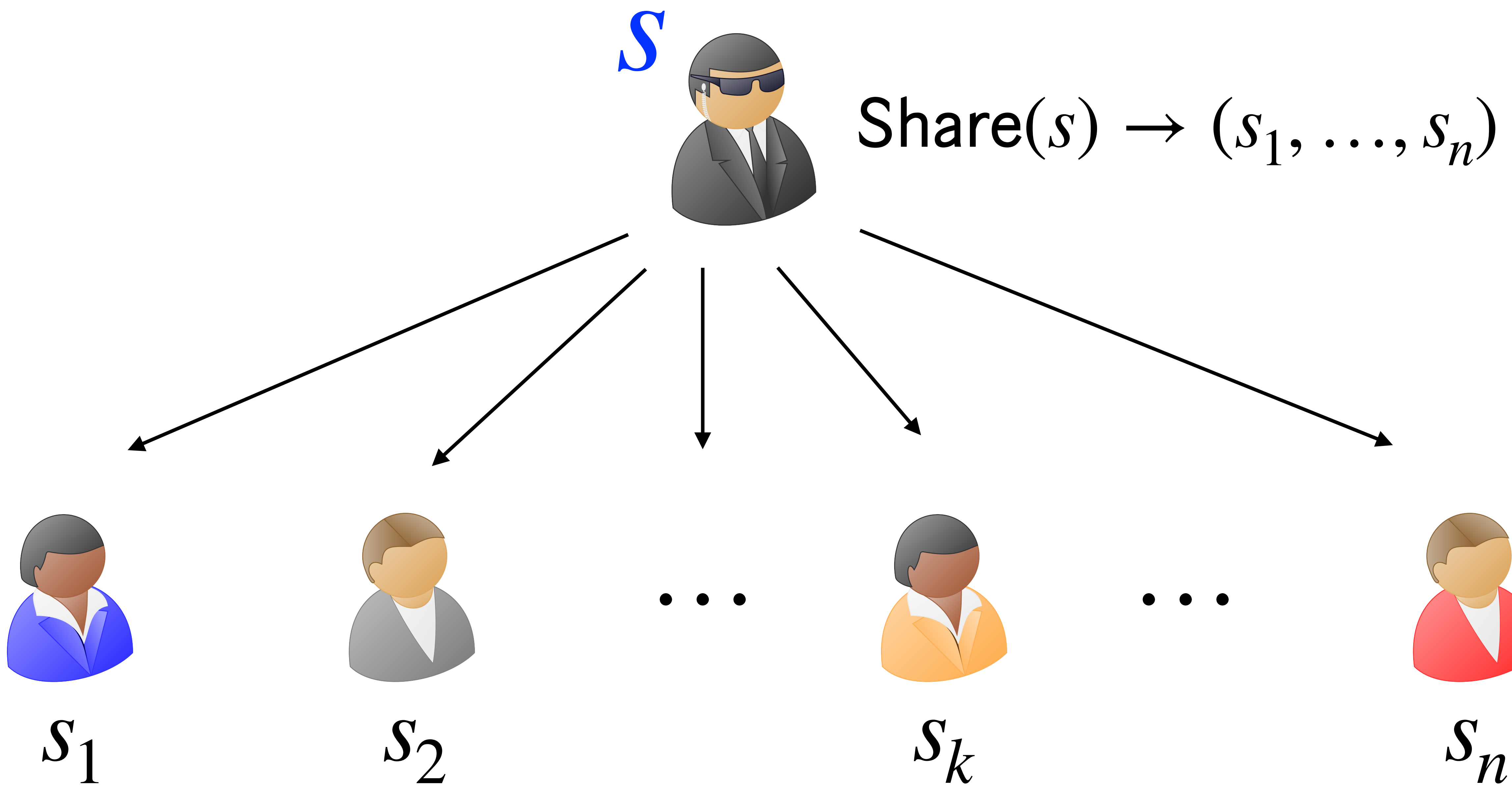

- Two algorithms
  - **Share:** Generate $n$ shares
  - **Recon:** Reconstruct secret from at least $k$ shares
- Properties
  - **Correctness:** Any subset of $\ell \geq k$ shares can be used to recover the secret
  - **Privacy:** Any subset of $\ell < k$ shares does not reveal anything about the secret

# $k$-out-of-$n$ Secret Sharing



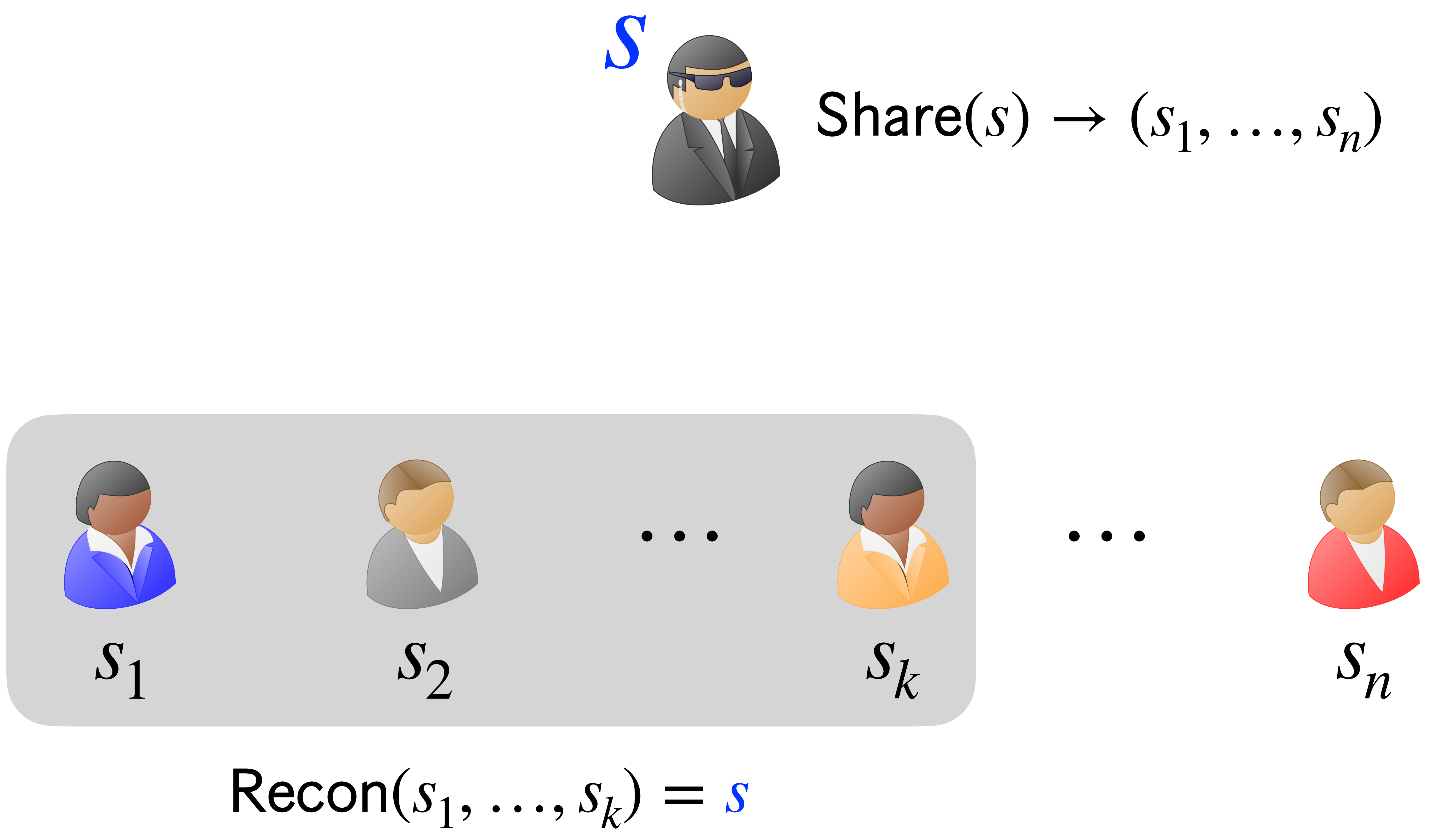

- Two algorithms
  - **Share:** Generate $n$ shares
  - **Recon:** Reconstruct secret from at least $k$ shares
- Properties
  - **Correctness:** Any subset of $\ell \geq k$ shares can be used to recover the secret
  - **Privacy:** Any subset of $\ell < k$ shares does not reveal anything about the secret

# $k$-out-of-$n$ Secret Sharing



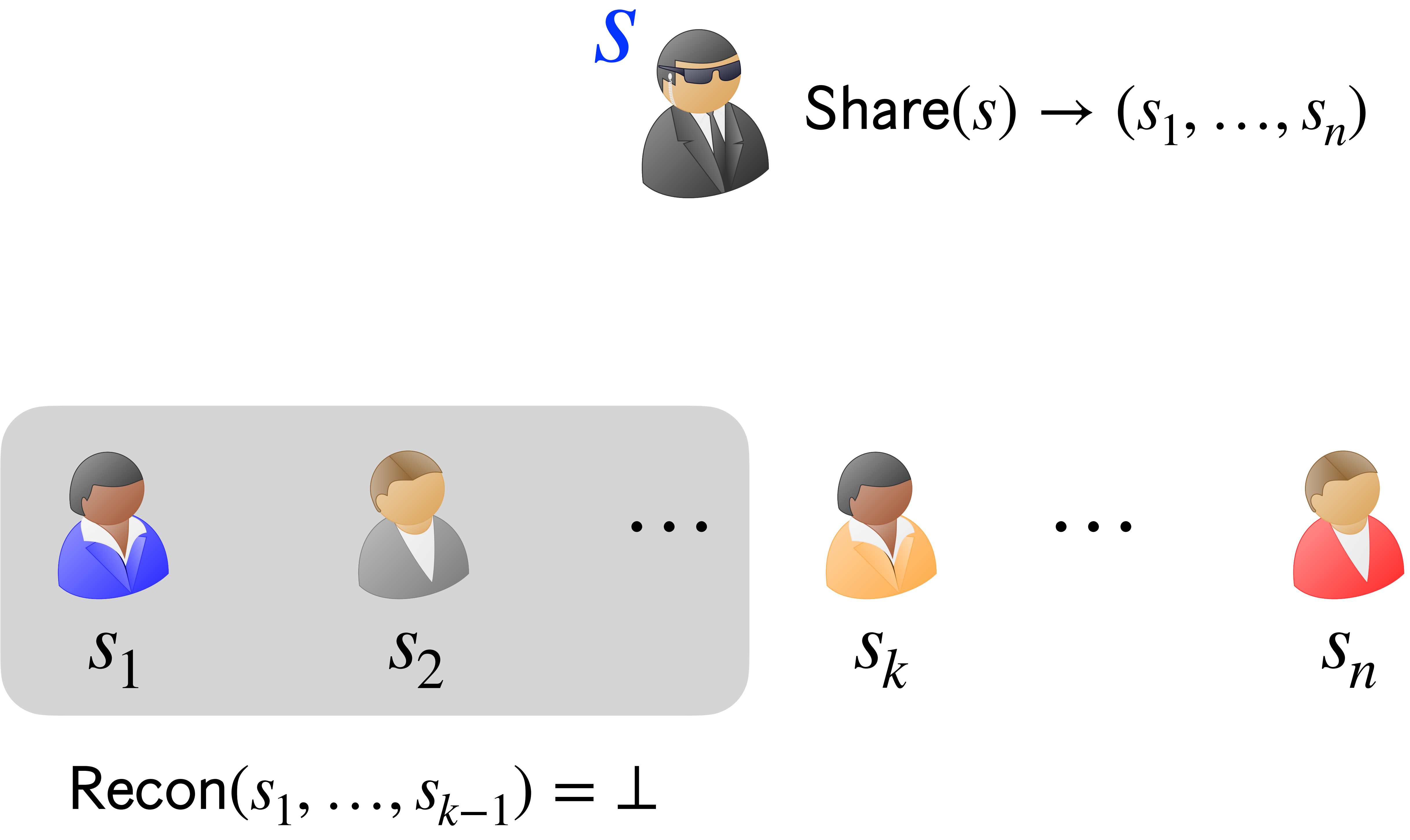

- Two algorithms
  - **Share:** Generate $n$ shares
  - **Recon:** Reconstruct secret from at least $k$ shares
- Properties
  - **Correctness:** Any subset of $\ell \geq k$ shares can be used to recover the secret
  - **Privacy:** Any subset of $\ell < k$ shares does not reveal anything about the secret

# $k$-out-of-$n$ Secret Sharing




- Two algorithms
  - **Share:** Generate $n$ shares
  - **Recon:** Reconstruct secret from at least $k$ shares
- Properties
  - **Correctness:** Any subset of $\ell \geq k$ shares can be used to recover the secret
  - **Privacy:** Any subset of $\ell < k$ shares does not reveal anything about the secret

# $k$-out-of-$n$ Secret Sharing




- Two algorithms
  - **Share:** Generate $n$ shares
  - **Recon:** Reconstruct secret from at least $k$ shares
- Properties
  - **Correctness:** Any subset of $\ell \geq k$ shares can be used to recover the secret
  - **Privacy:** Any subset of $\ell < k$ shares does not reveal anything about the secret

# 2-out-of-2 Secret Sharing

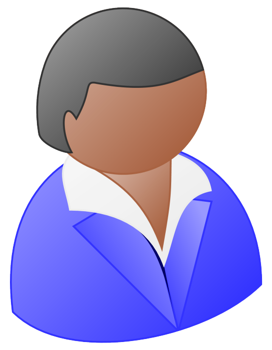
$s \in \{0,1\}^m$

# 2-out-of-2 Secret Sharing

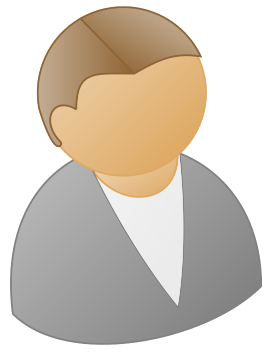
$s \in \{0,1\}^m$

$s_1 \leftarrow \{0,1\}^m$

$s_2 = s \oplus s_1$



$s_1$



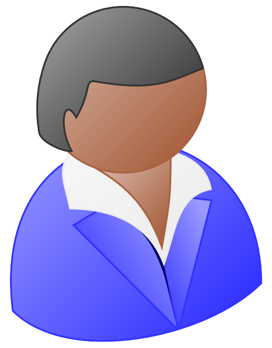
$s_2$

# 2-out-of-2 Secret Sharing

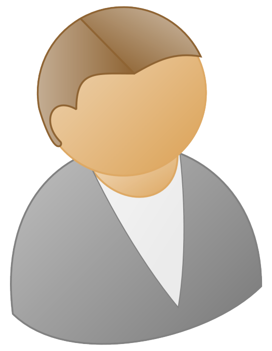
$s \in \{0,1\}^m$

$s_1 \leftarrow \{0,1\}^m$

$s_2 = s \oplus s_1$



$s_1$



$s_2$

$\forall v_1, v_2, r \in \{0,1\}^m$

$$P[s_1 = r \mid s = v_1] = P[s_1 = r \mid s = v_2]$$

$$P[s_2 = r \mid s = v_1] = P[s_2 = r \mid s = v_2]$$

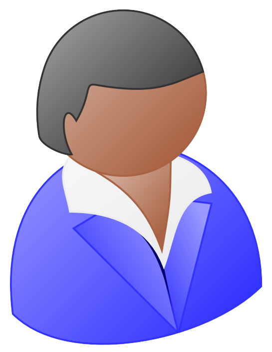
- Security
  - A party’s share taking on a particular value is equally likely for every secret
  - Similar to One-time Pads
- Reconstruction
  - Simply XOR the shares

# 2-out-of-2 Secret Sharing

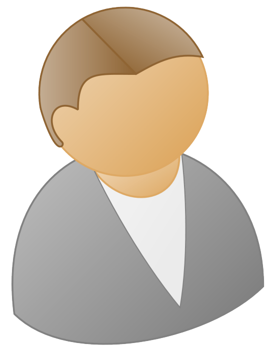
$s \in \{0,1\}^m$

$s_1 \leftarrow \{0,1\}^m$

$s_2 = s \oplus s_1$



$s_1$



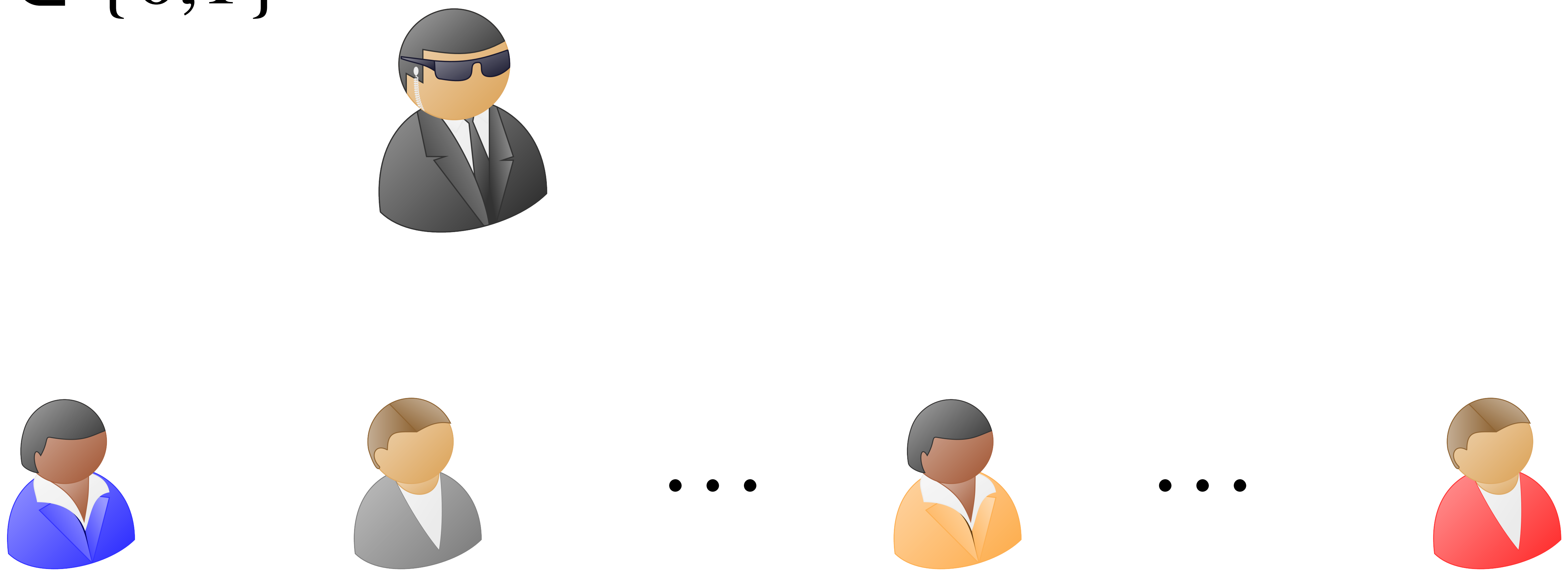
$s_2$

$\text{Recon}(s_1, s_2) = s_1 \oplus s_2 = s$

- Security
  - A party's share taking on a particular value is equally likely for every secret
  - Similar to One-time Pads
- Reconstruction
  - Simply XOR the shares

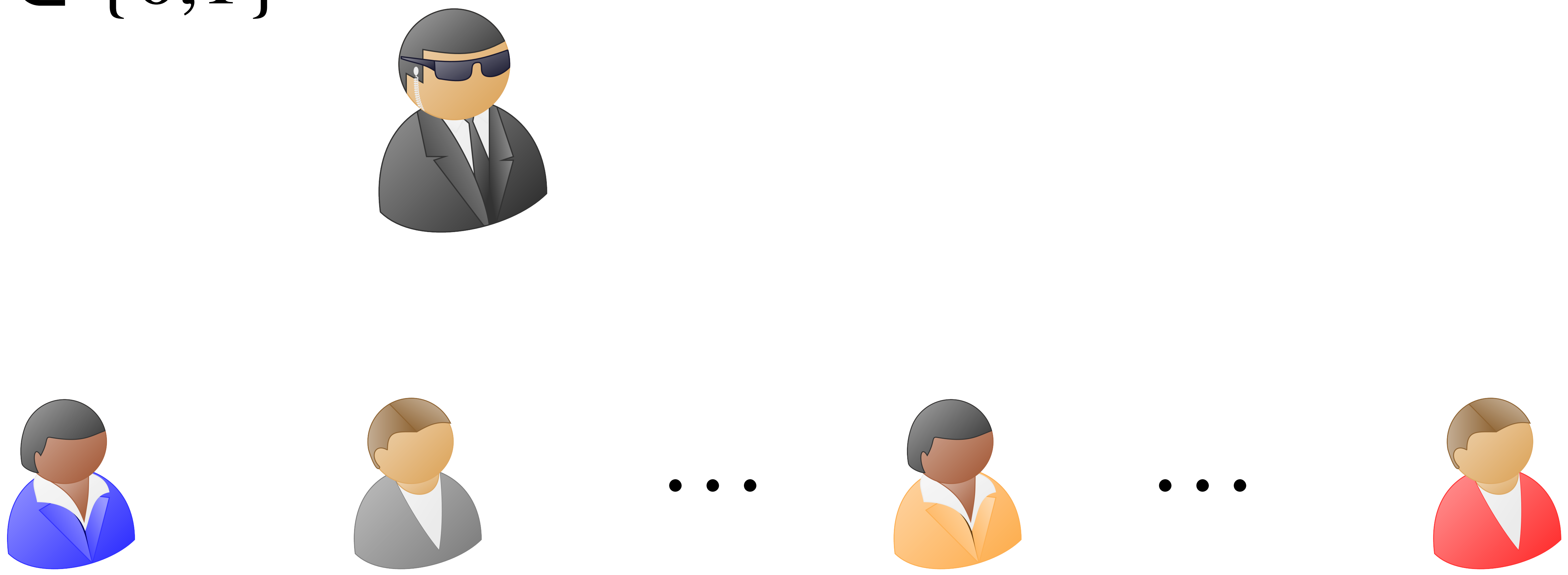
# $n$-out-of-$n$ Secret Sharing

$s \in \{0,1\}^m$



# $n$-out-of-$n$ Secret Sharing

$s \in \{0,1\}^m$



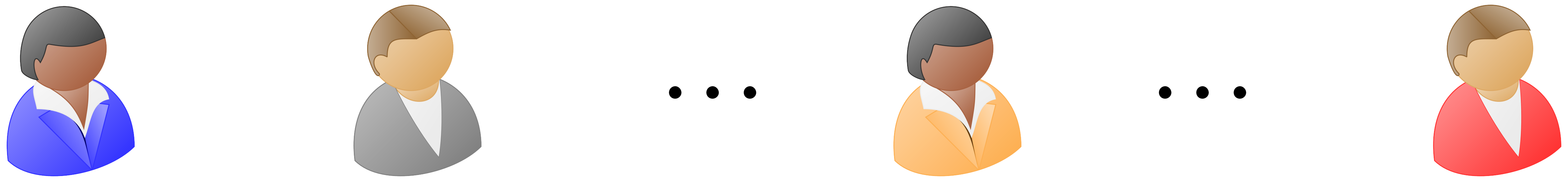
$s_1$ $s_2$ $\cdots$ $s_k$ $\cdots$ $s_n$

# $n$-out-of-$n$ Secret Sharing

$s \in \{0,1\}^m$

$s_1, \ldots, s_{n-1} \leftarrow \{0,1\}^m$

$s_n = s \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_{n-1}$



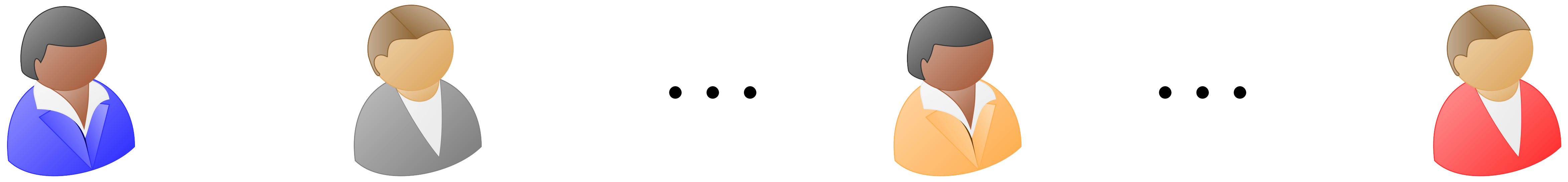
$s_1$ $s_2$ $\ldots$ $s_k$ $\ldots$ $s_n$

# $n$-out-of-$n$ Secret Sharing

$s \in \{0,1\}^m$

$s_1, \ldots, s_{n-1} \leftarrow \{0,1\}^m$

$s_n = s \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_{n-1}$



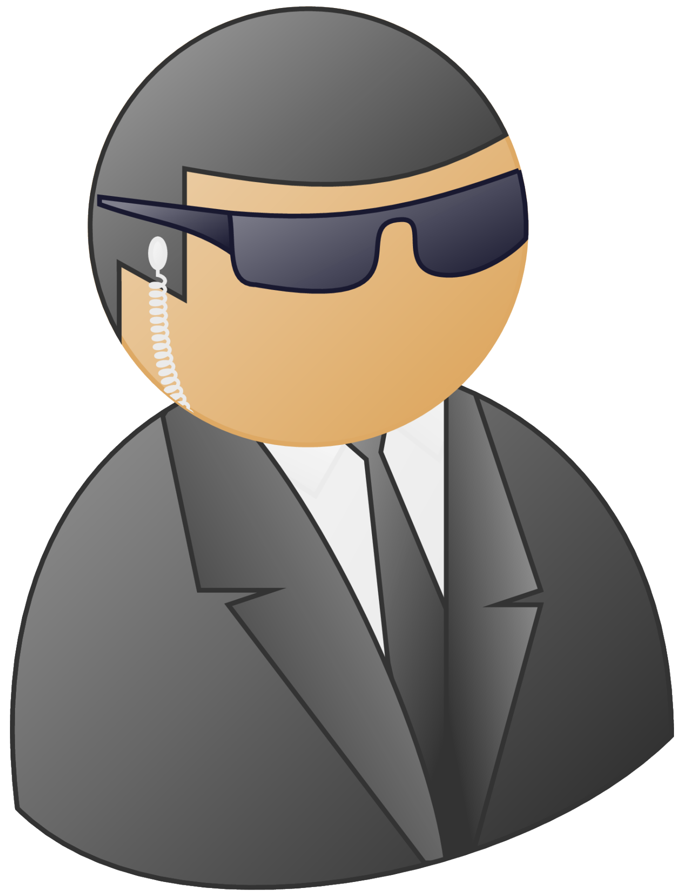
$s_1$ $\quad$ $s_2$ $\quad \cdots \quad$ $s_k$ $\quad \cdots \quad$ $s_n$

$\mathsf{Recon}(s_1, \ldots, s_n) = s_1 \oplus s_2 \oplus \ldots \oplus s_n = s$

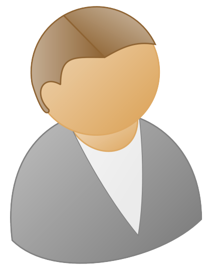
- Security
  - The distribution of any subset of $n-1$ shares is independent of the secret
- Reconstruction
  - Simply XOR the shares

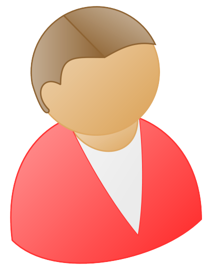
# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

3-out-of-4 Secret Sharing



$s \in \{0,1\}^m$

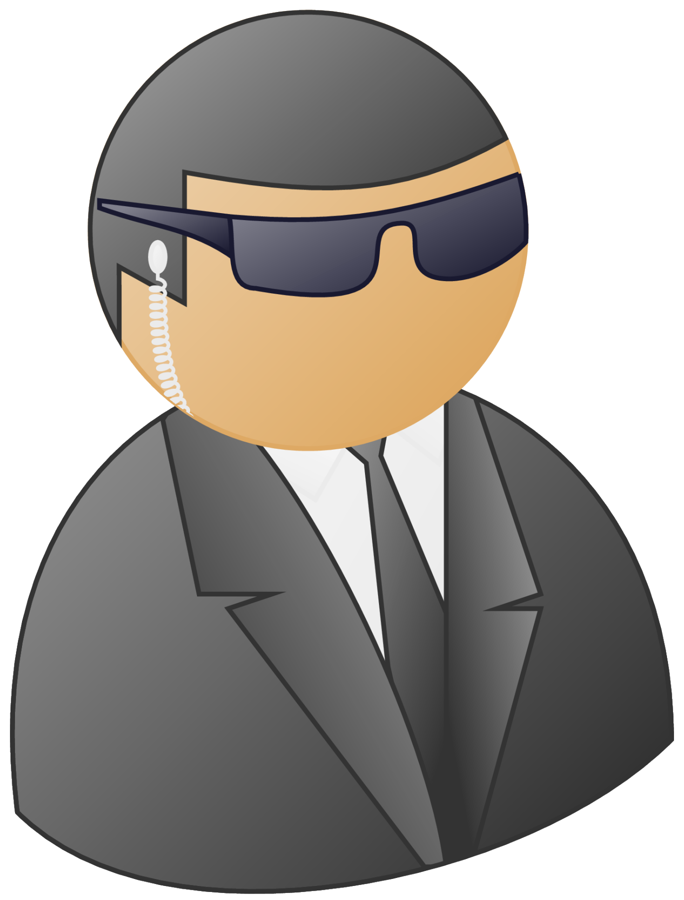





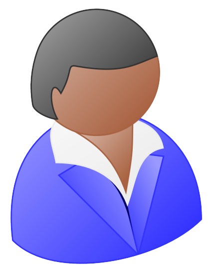
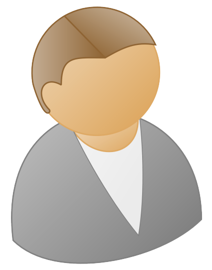
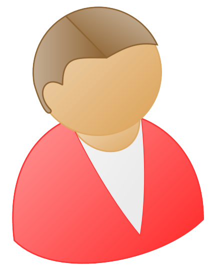
# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing



$s \in \{0,1\}^m$

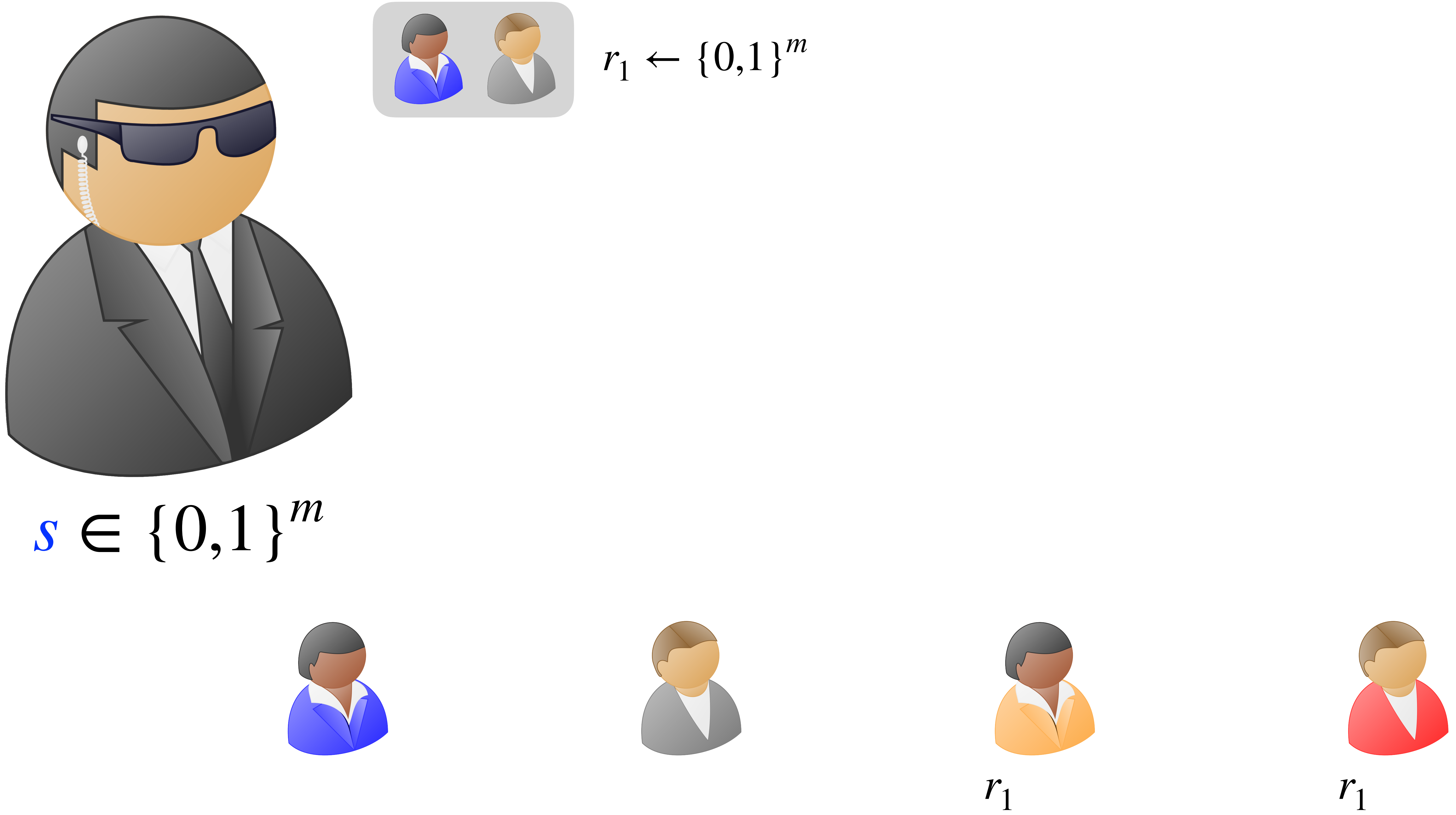
   

- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

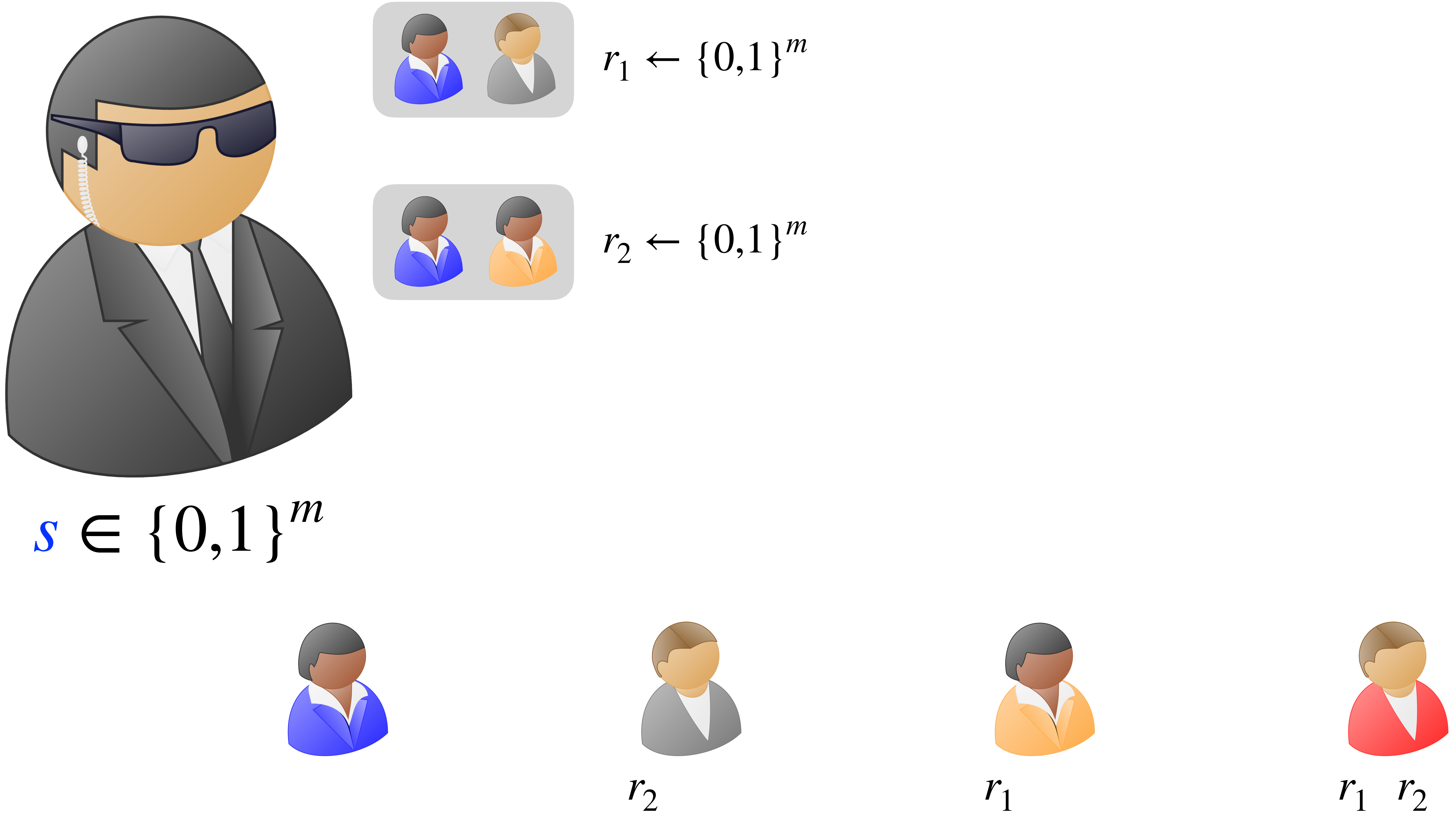
### 3-out-of-4 Secret Sharing




- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

3-out-of-4 Secret Sharing

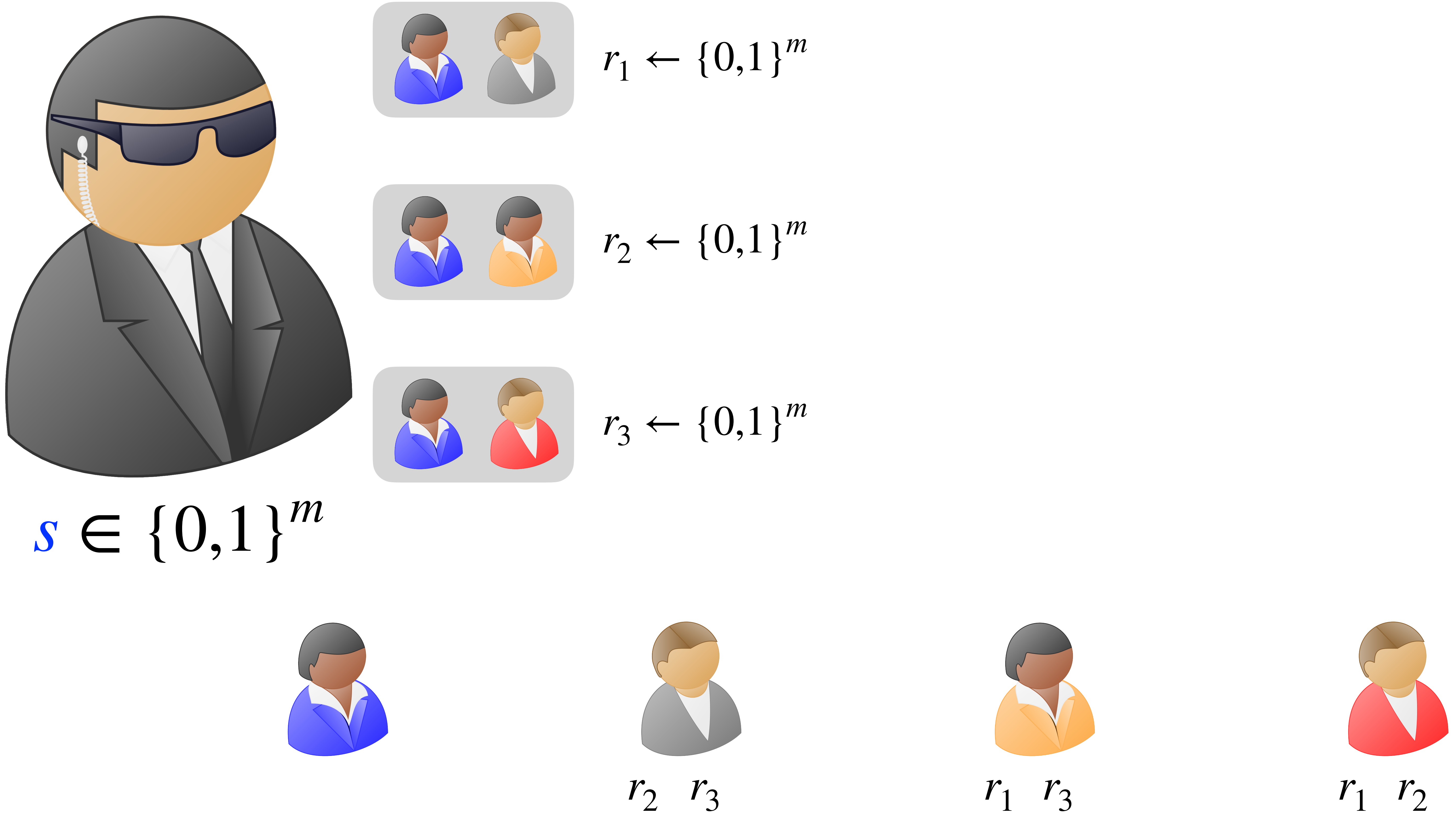



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

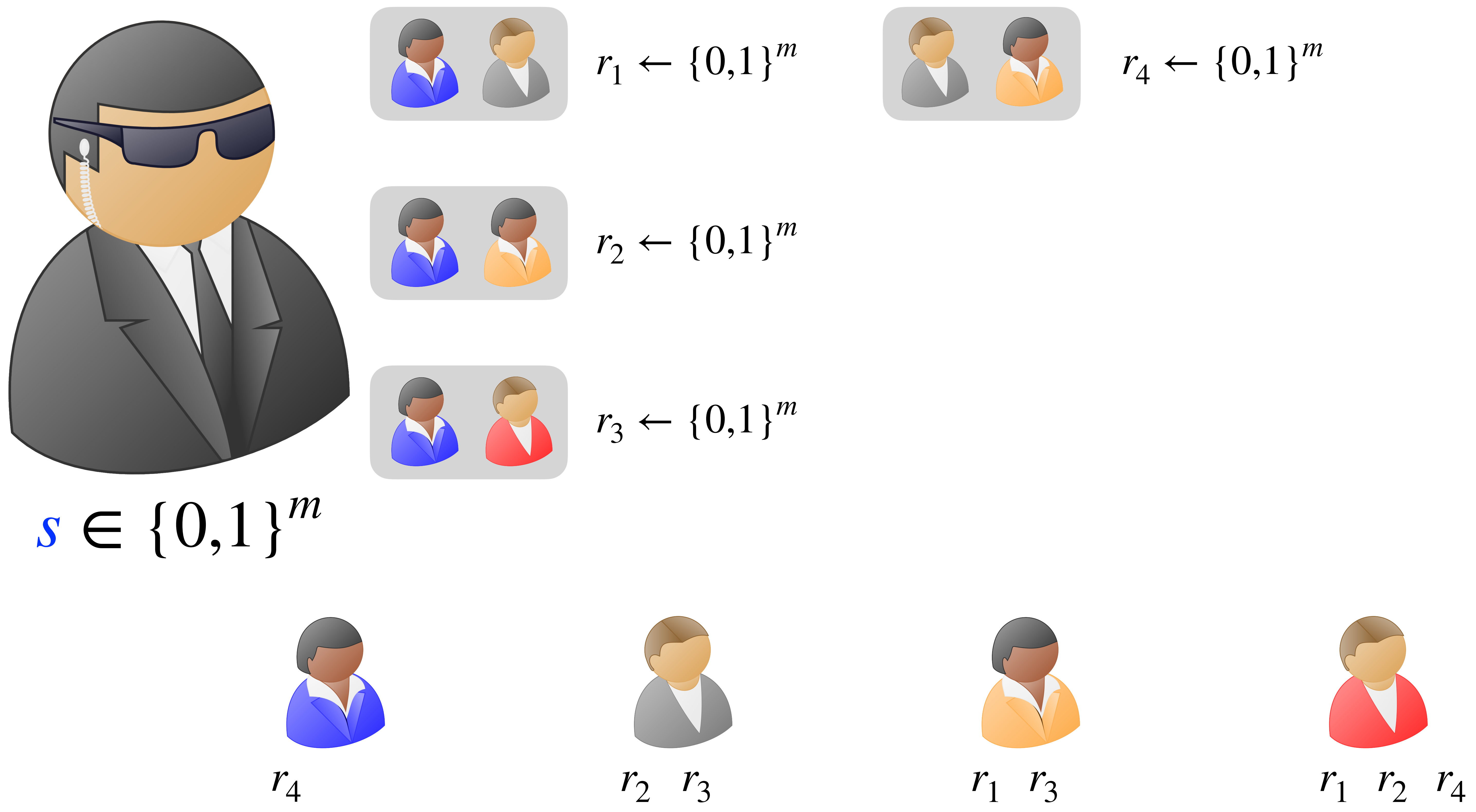



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

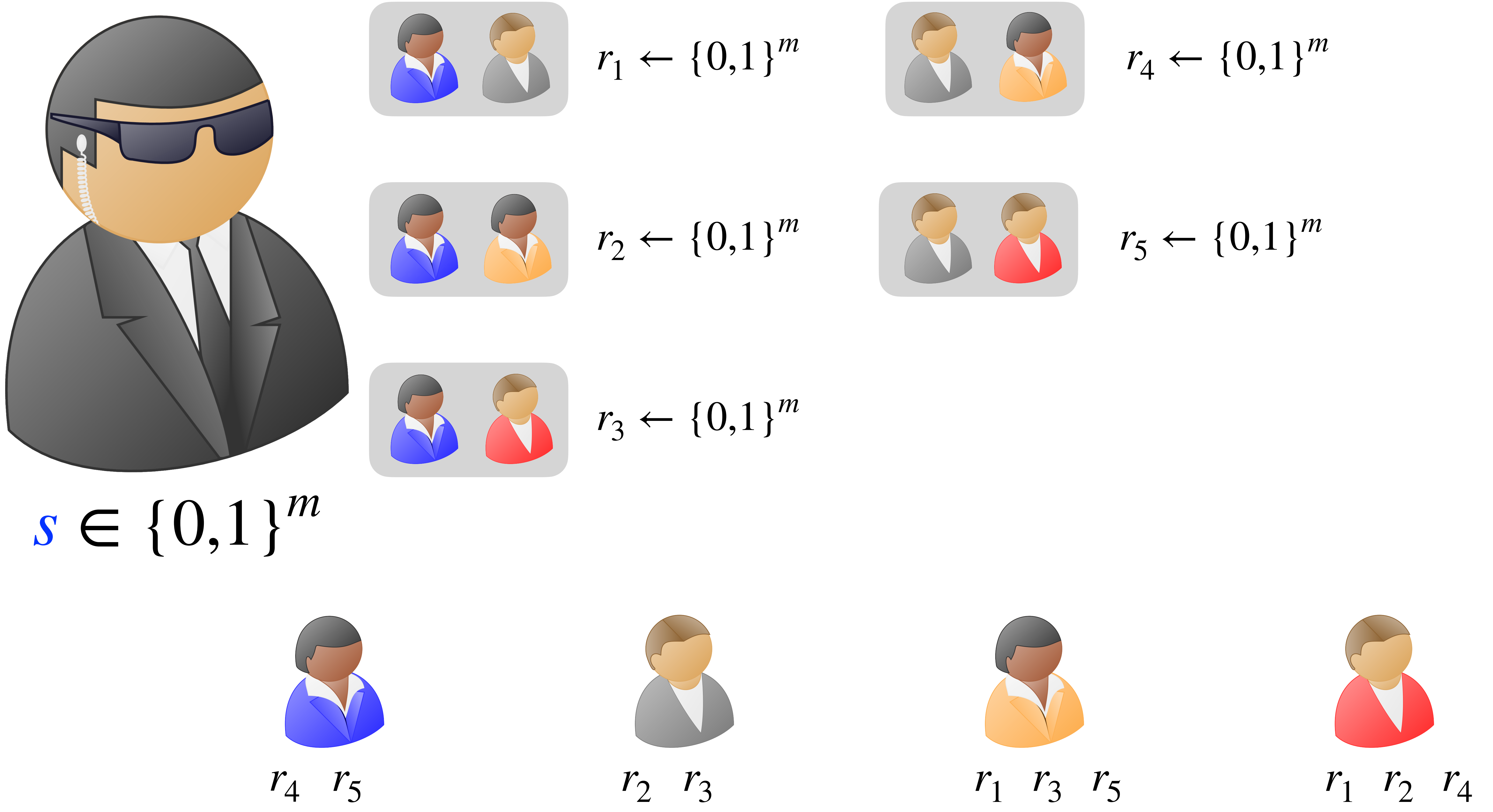



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\}\backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

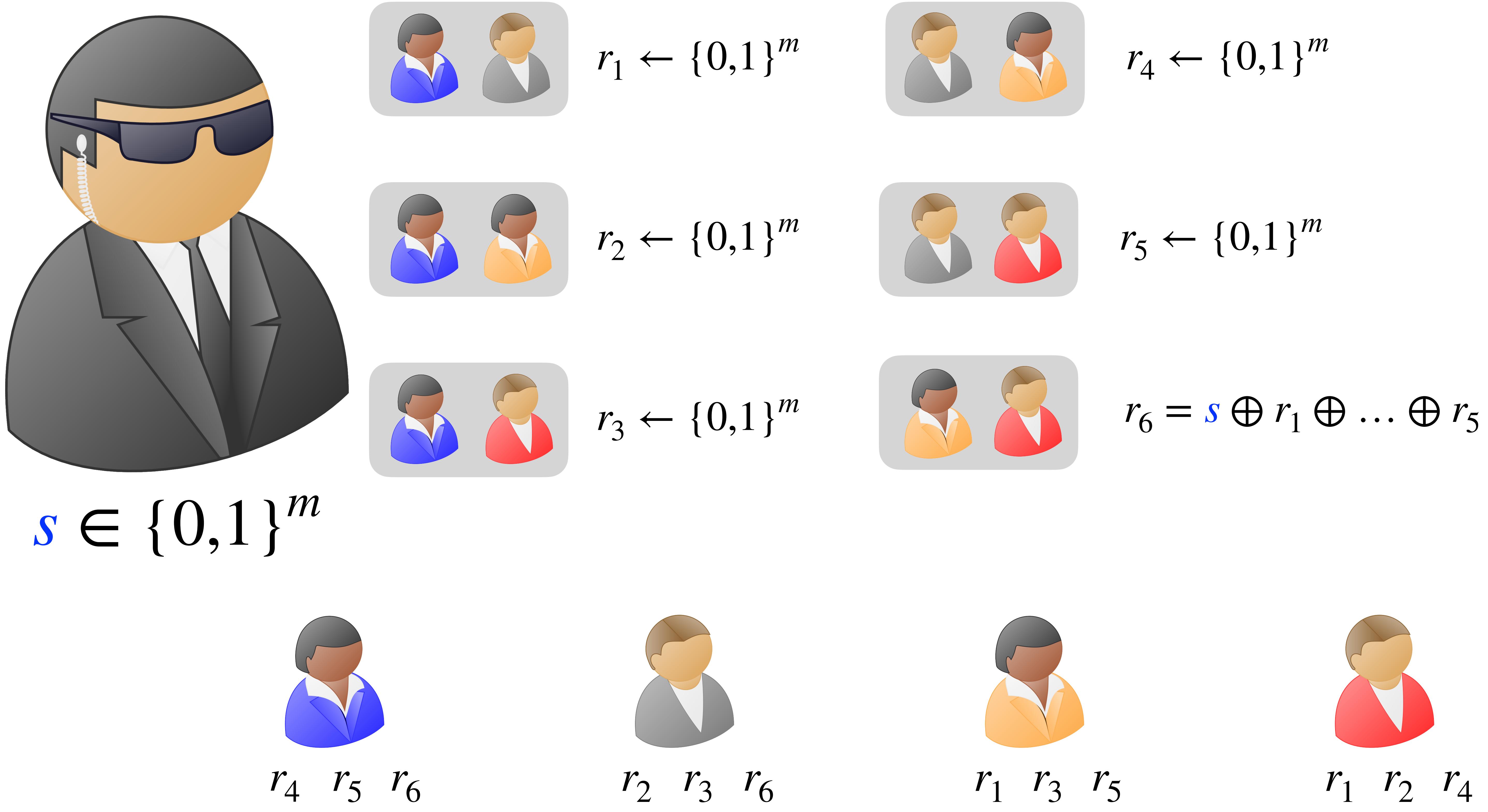



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

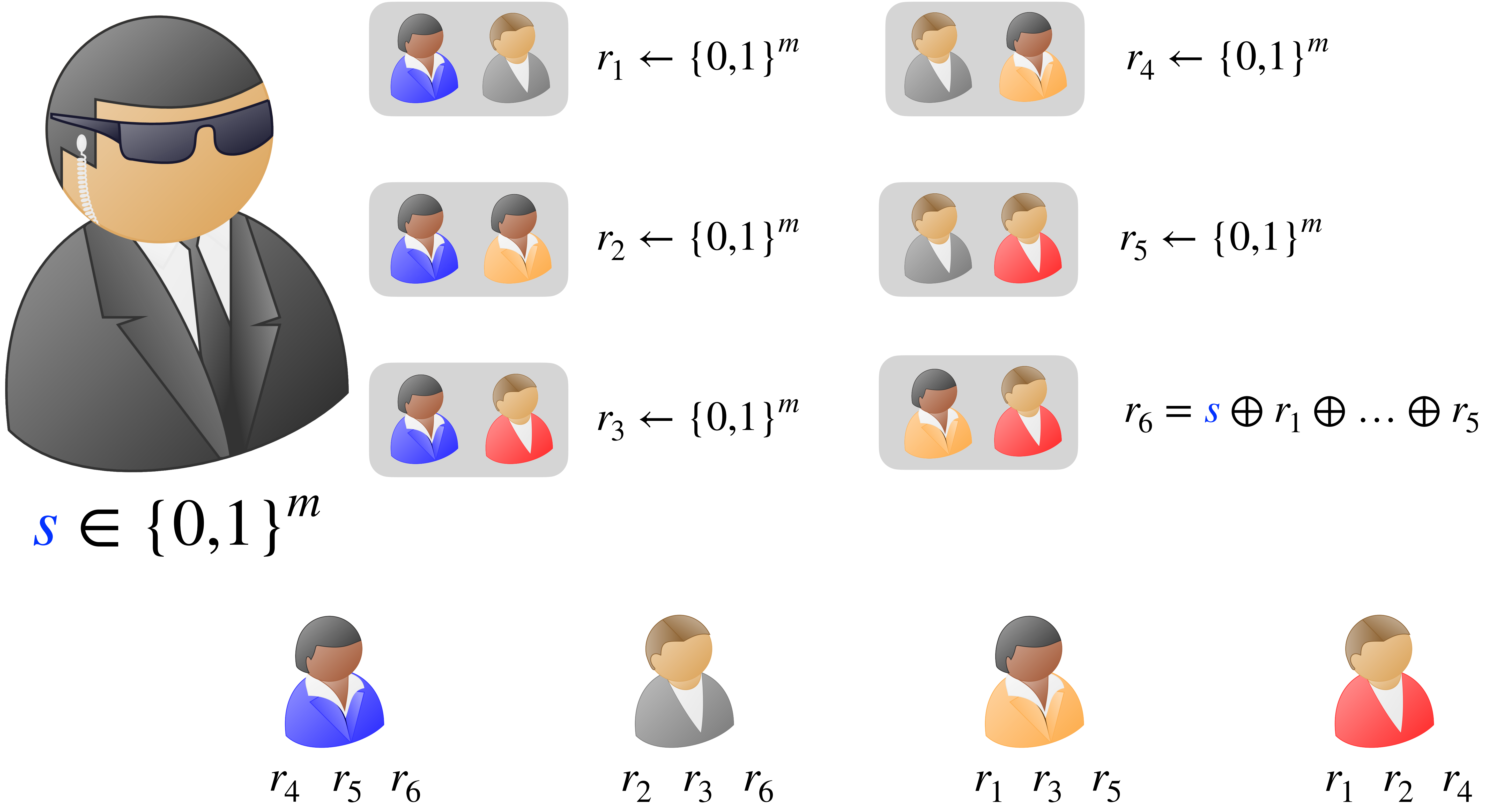



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

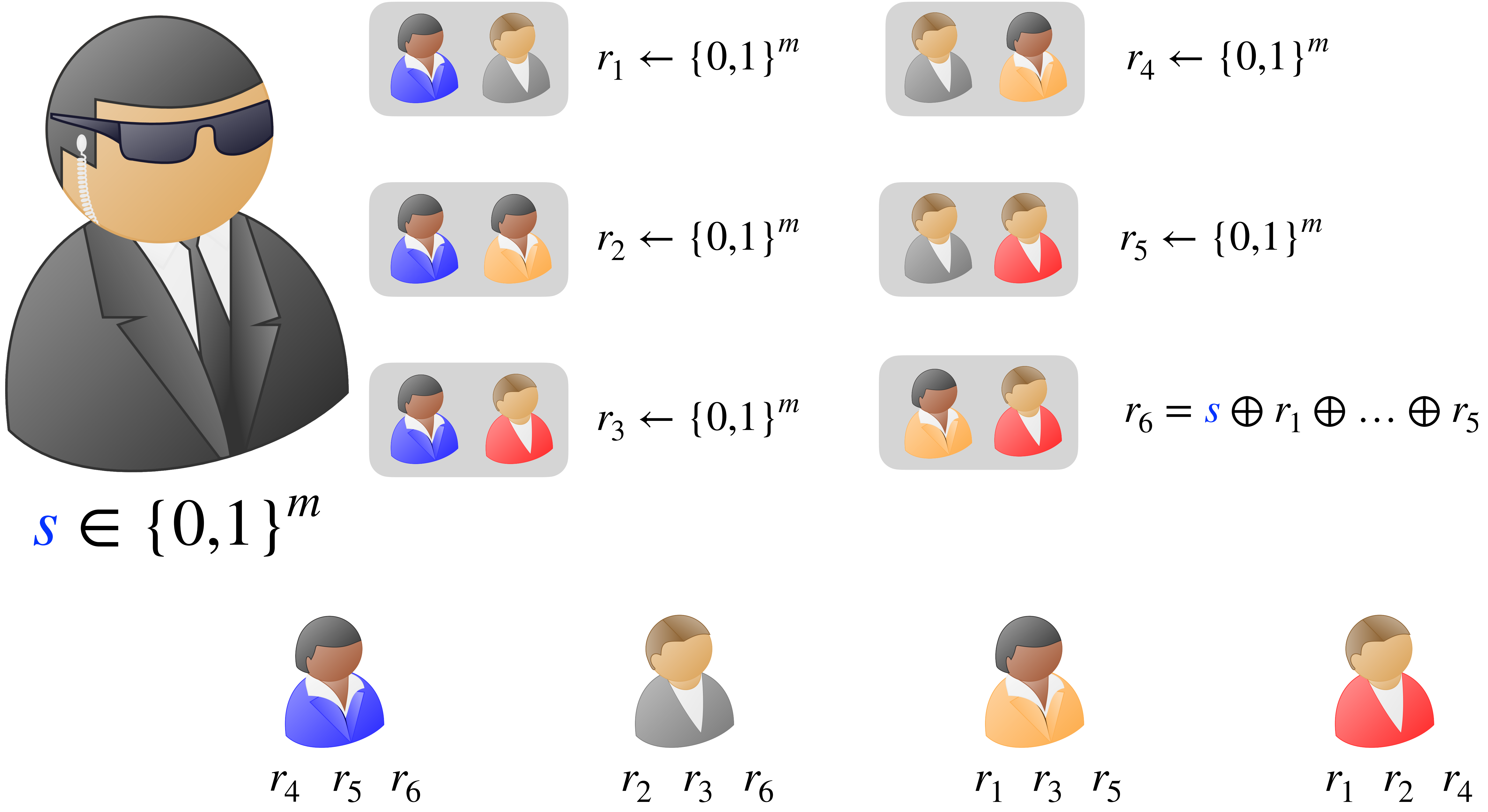



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$
- Correctness
  - Any $k$ sized subset of parties covers all $r_i$ values
  - Reconstruction Algorithm: XOR all $r_i$ values

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

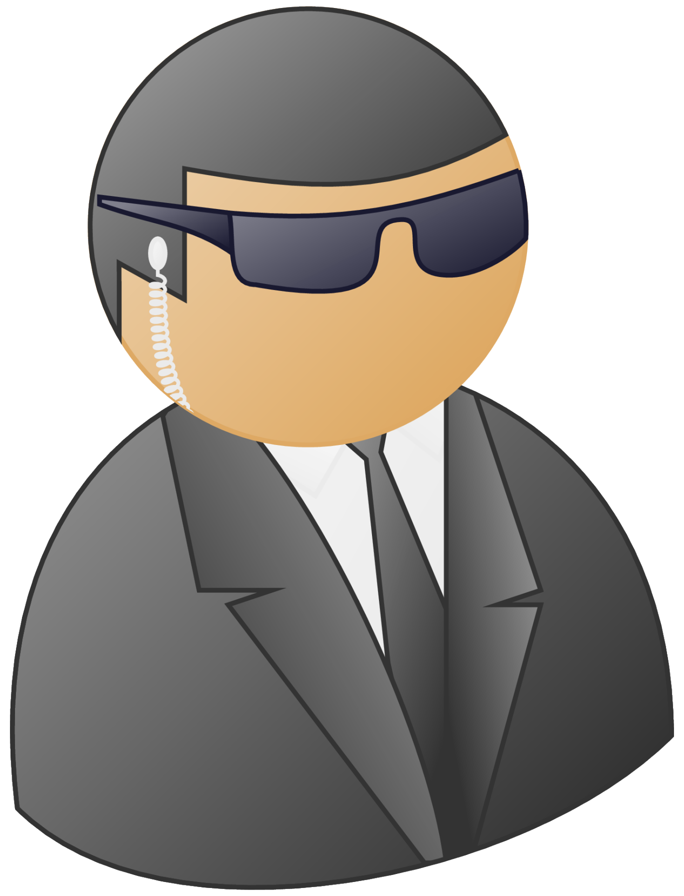
3-out-of-4 Secret Sharing




- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$
- Correctness
  - Any $k$ sized subset of parties covers all $r_i$ values
  - Reconstruction Algorithm: XOR all $r_i$ values
- Privacy
  - Any $k-1$ size subset of parties does not have at least one $r_i$ value

# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

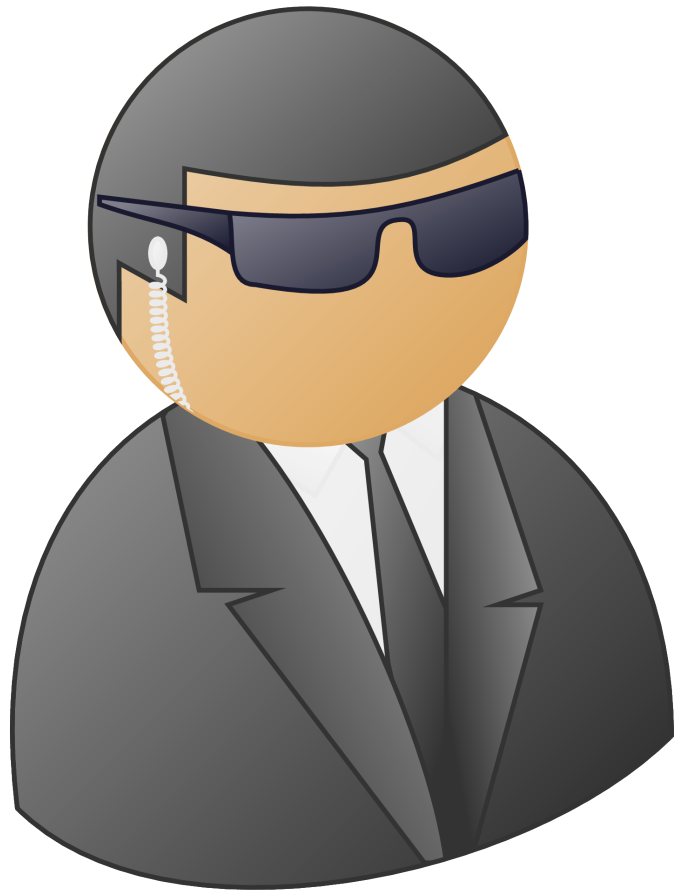
3-out-of-4 Secret Sharing



$s \in \{0,1\}^m$

# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

3-out-of-4 Secret Sharing

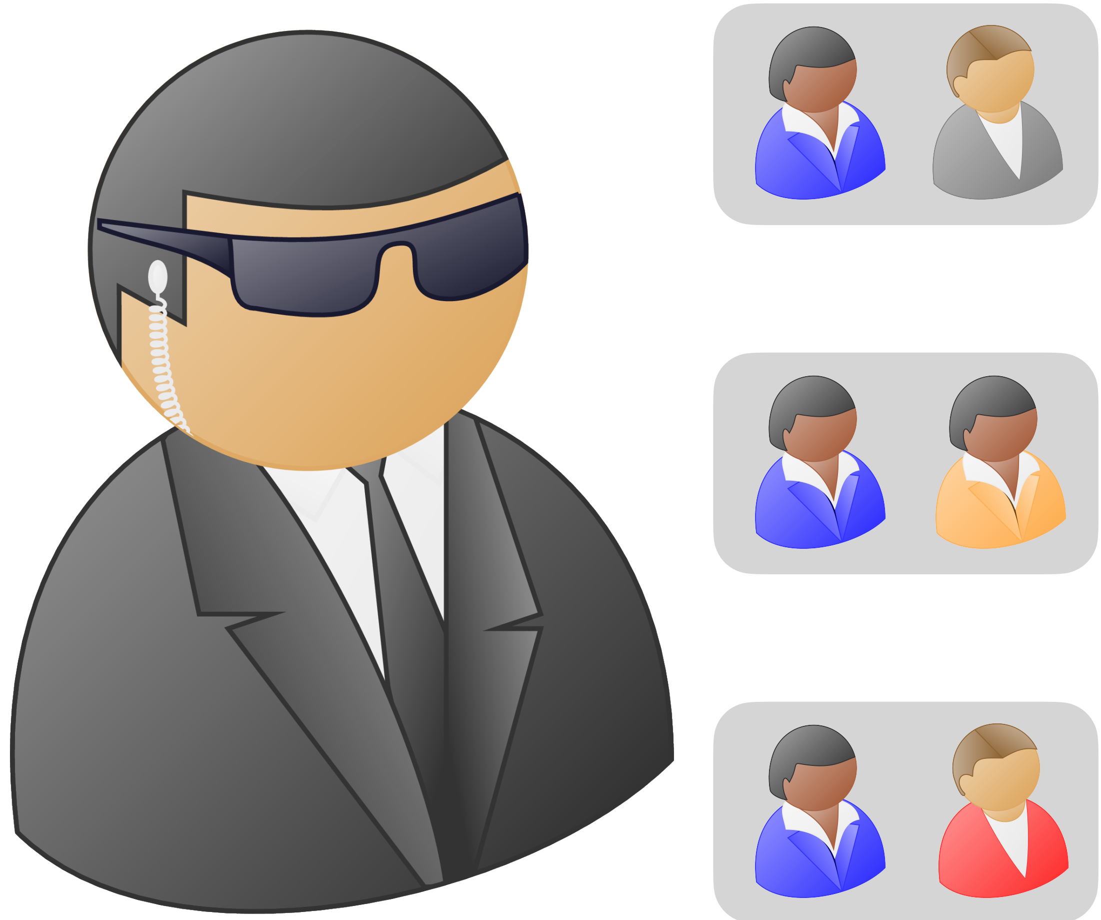


$s \in \{0,1\}^m$

- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

3-out-of-4 Secret Sharing



$r_1 \leftarrow \{0,1\}^m$

$r_2 \leftarrow \{0,1\}^m$

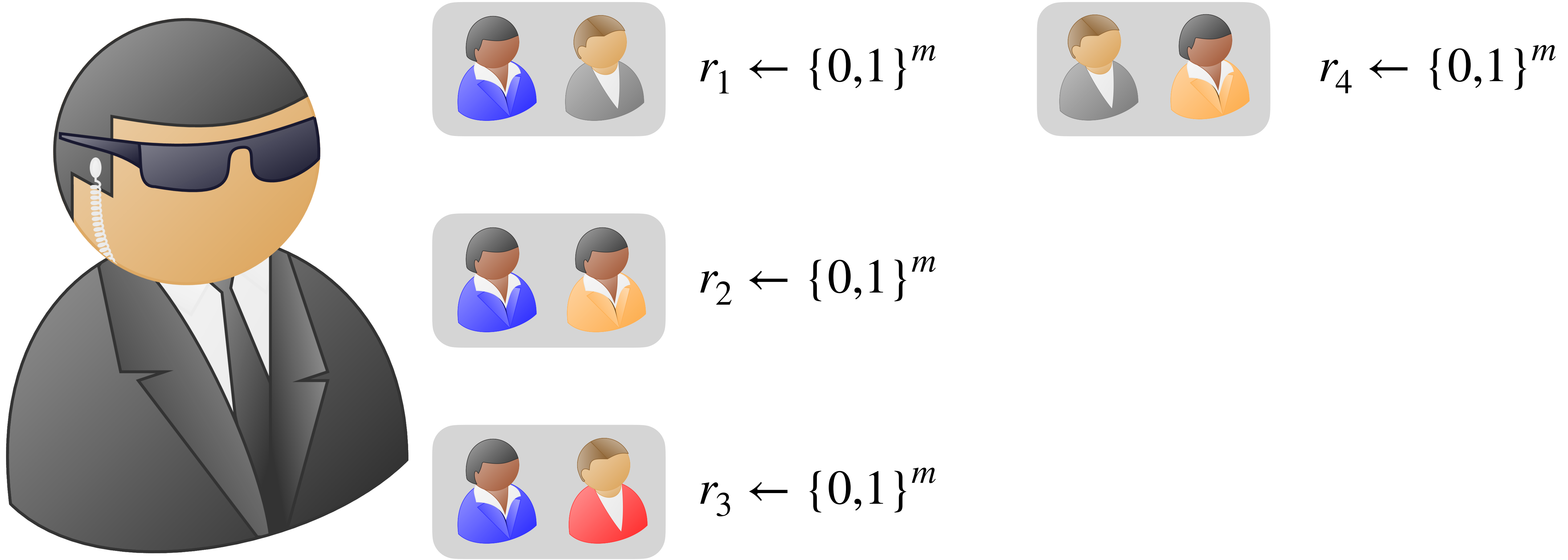
$r_3 \leftarrow \{0,1\}^m$

$s \in \{0,1\}^m$

- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

3-out-of-4 Secret Sharing

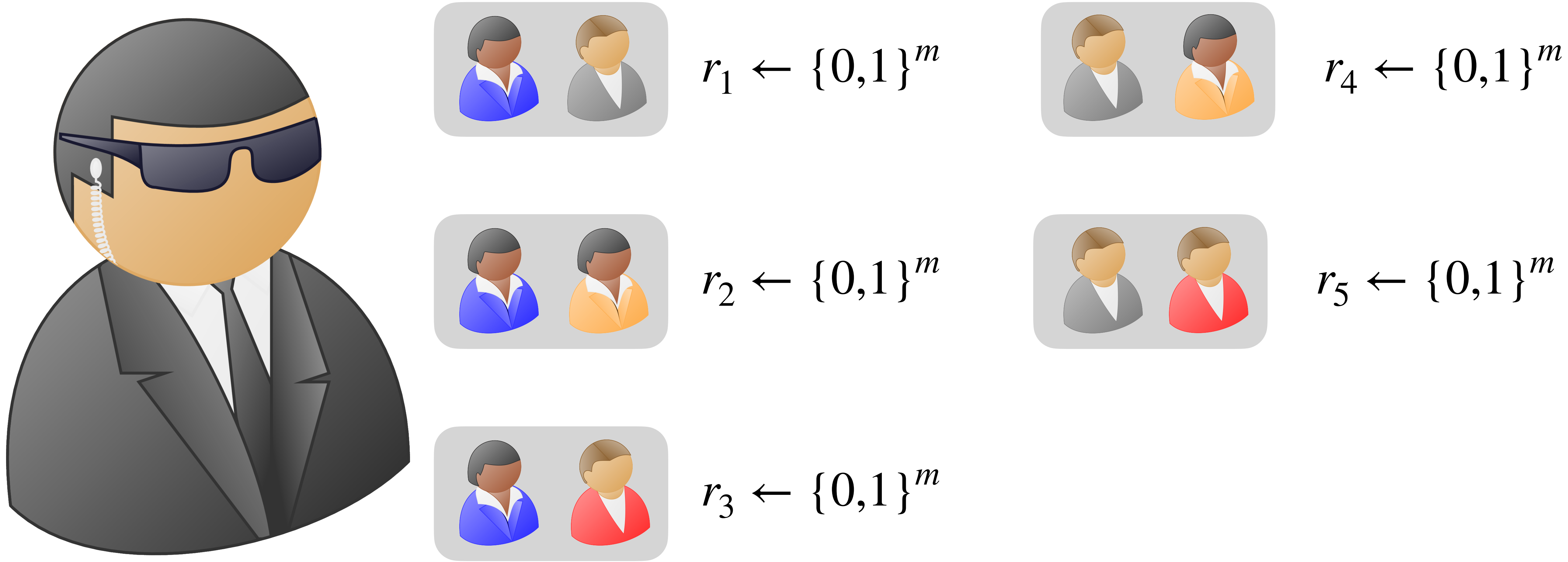



$s \in \{0,1\}^m$

- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\}\setminus S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing
## Combinatorial Construction

3-out-of-4 Secret Sharing



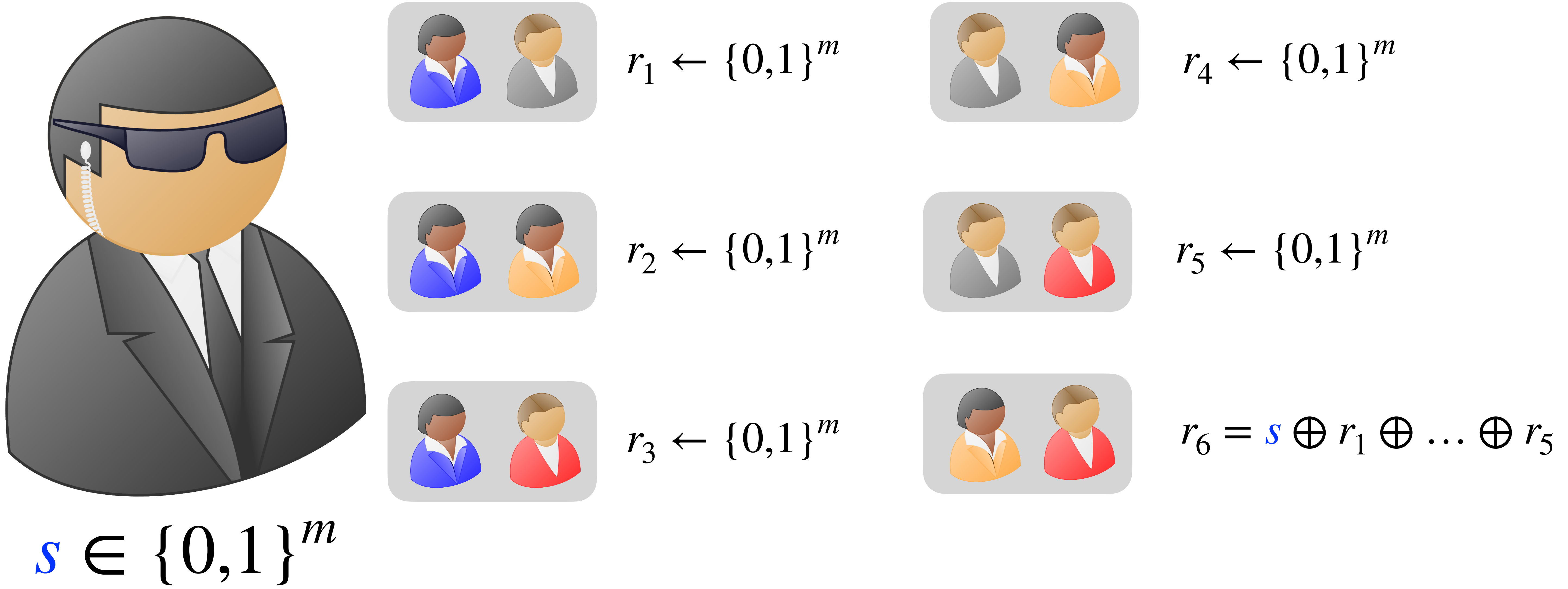

$s \in \{0,1\}^m$

- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

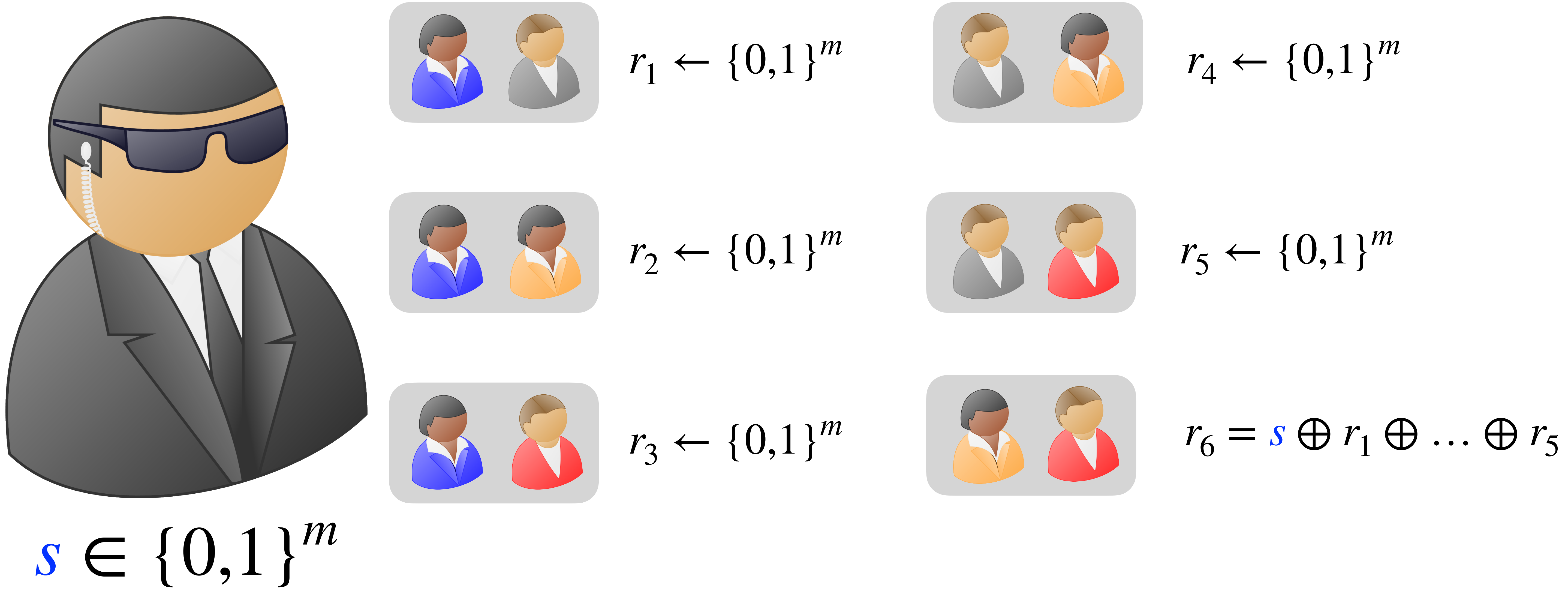



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\}\backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

### 3-out-of-4 Secret Sharing

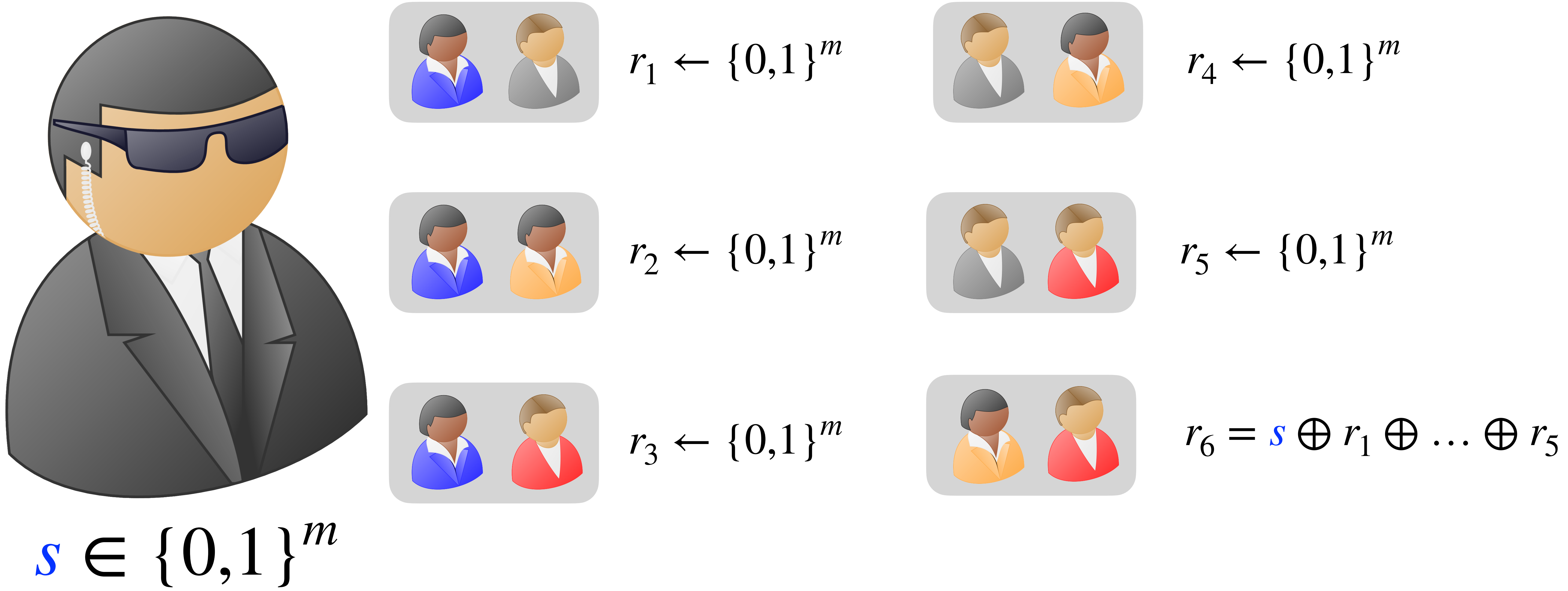



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\}\backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$
- Correctness
  - Any $k$ sized subset of parties covers all $r_i$ values
  - Reconstruction Algorithm: XOR all $r_i$ values

# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

3-out-of-4 Secret Sharing

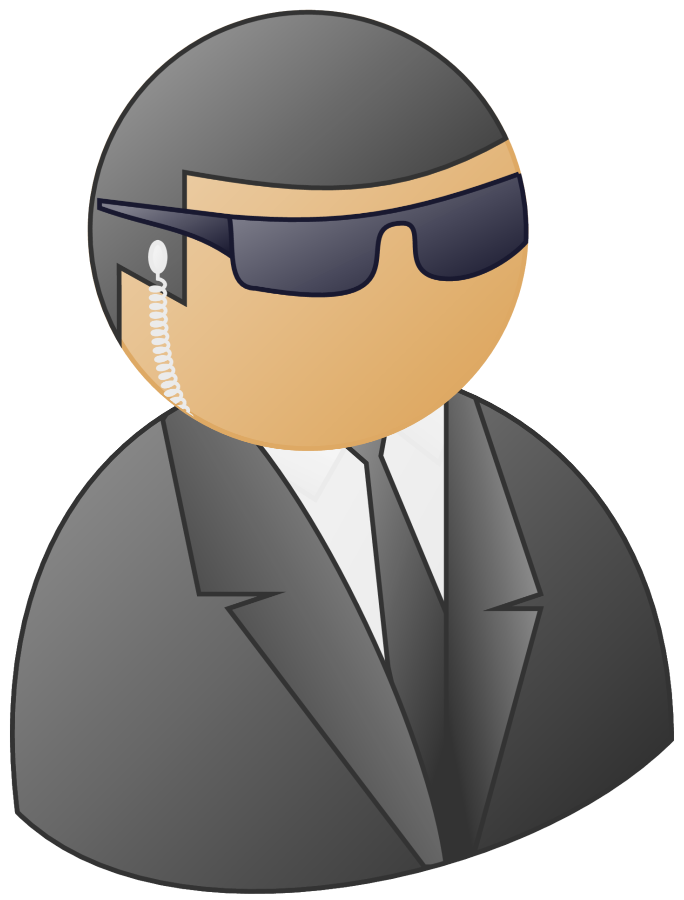



- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$
- Correctness
  - Any $k$ sized subset of parties covers all $r_i$ values
  - Reconstruction Algorithm: XOR all $r_i$ values
- Privacy
  - Any $k-1$ size subset of parties does not have at least one $r_i$ value

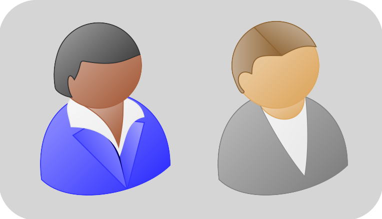
# $k$-out-of-$n$ Secret Sharing

## Combinatorial Construction

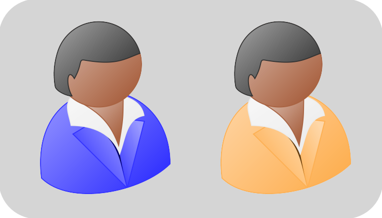
### 3-out-of-4 Secret Sharing



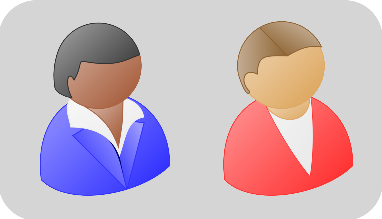
$s \in \{0,1\}^m$



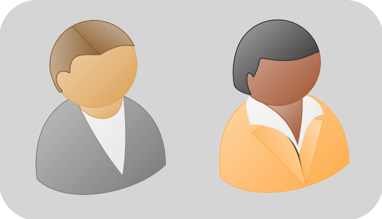
$r_1 \leftarrow \{0,1\}^m$



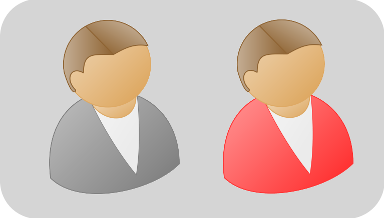
$r_2 \leftarrow \{0,1\}^m$



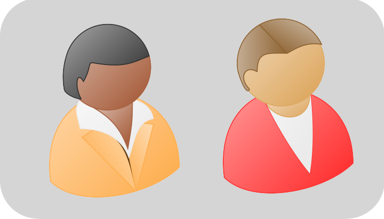
$r_3 \leftarrow \{0,1\}^m$



$r_4 \leftarrow \{0,1\}^m$



$r_5 \leftarrow \{0,1\}^m$



$r_6 = s \oplus r_1 \oplus \ldots \oplus r_5$

- Running time of Share
  - Proportional to number of $k-1$ subsets
  - $\binom{n}{k}$ number of shares — combinatorial blow-up!

- Share Algorithm
  - For each subset $S_i \subset \{1,\ldots,n\}$ of $k-1$ parties sample a random value $r_i$
  - Give $r_i$ to the parties **NOT** in $S_i$ i.e., parties in $\{1,\ldots,n\} \backslash S_i$
  - For the last subset $S_L$, set $r_L = s \oplus r_1 \oplus \ldots \oplus r_{L-1}$
- Correctness
  - Any $k$ sized subset of parties covers all $r_i$ values
  - Reconstruction Algorithm: XOR all $r_i$ values
- Privacy
  - Any $k-1$ size subset of parties does not have at least one $r_i$ value

# Arithmetic Over $\mathbb{Z}_p$

- $p$ is a prime number
- Recall: $\mathbb{Z}_p$ is an (abelian) group
  - Can add and subtract over $\mathbb{Z}_p$. Subtraction is equivalent to adding the “additive inverse” i.e., inverse of the element over the group $\mathbb{Z}_p$
- Can generalize previous schemes to work over $\mathbb{Z}_p$
- In fact, the XOR based scheme, with $m = 1$, are equivalent to working over $\mathbb{Z}_2$

# Arithmetic Over $\mathbb{Z}_p$

- $p$ is a prime number
- Recall: $\mathbb{Z}_p$ is an (abelian) group
  - Can add and subtract over $\mathbb{Z}_p$. Subtraction is equivalent to adding the "additive inverse" i.e., inverse of the element over the group $\mathbb{Z}_p$
- Recall: $\mathbb{Z}_p^*$ is an (abelian) group
  - Can multiply and divide with any **non-zero** element in $\mathbb{Z}_p$. Division is equivalent to multiplying with the inverse of the element over the group $\mathbb{Z}_p^*$

# Polynomials Over $\mathbb{Z}_p$

- A degree-$d$ polynomial $p(x)$, where $d \geq 0$ is an integer, is of the form

$$p(x) = \sum_{i=0}^{d} c_i \cdot x^i = c_0 \ + \ c_1 \cdot x \ + \ \dots \ + \ c_d \cdot x^d$$

  where $c_0, \dots, c_d \in \mathbb{Z}_p$.

- $c_i$ is called the co-efficient of $x^i$; $c_0$ is called the constant co-efficient.

- **Evaluation:** For any $\alpha \in \mathbb{Z}_p$, $p(\alpha)$ is defined as

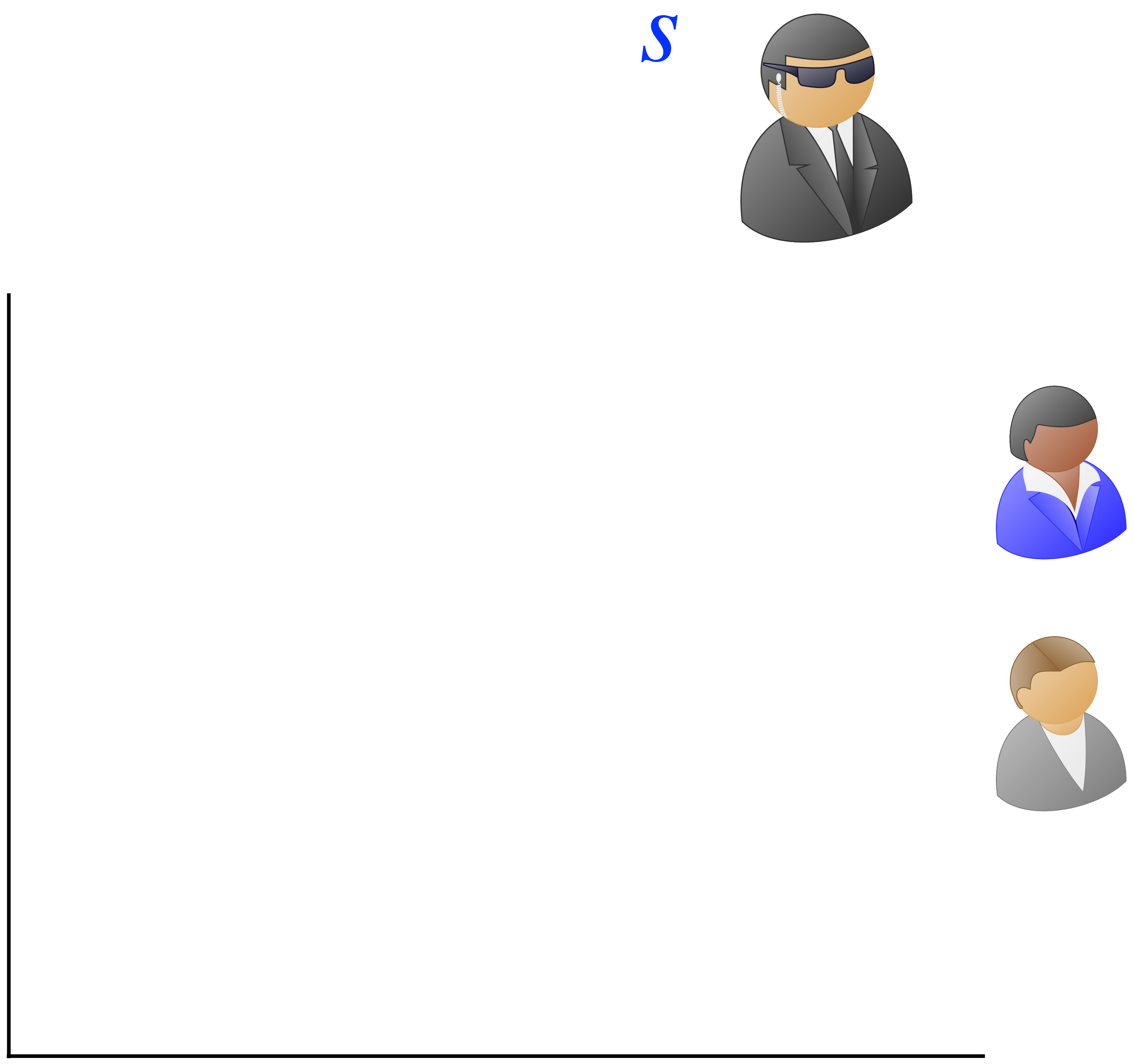
$$p(\alpha) = \sum_{i=0}^{d} c_i \cdot \alpha^i \bmod p.$$

- **Theorem:** Let $d \geq 0$ be any integer and let $\alpha_1, \dots, \alpha_{d+1} \in \mathbb{Z}_p$ be distinct. For all $\beta_1, \dots, \beta_{d+1} \in \mathbb{Z}_p$, there exists a **unique** degree-$d$ polynomial $p(x)$ such that

$$p(\alpha_i) = \beta_i \qquad \forall i \in \{1, \dots, d+1\}.$$

# Shamir Secret Sharing

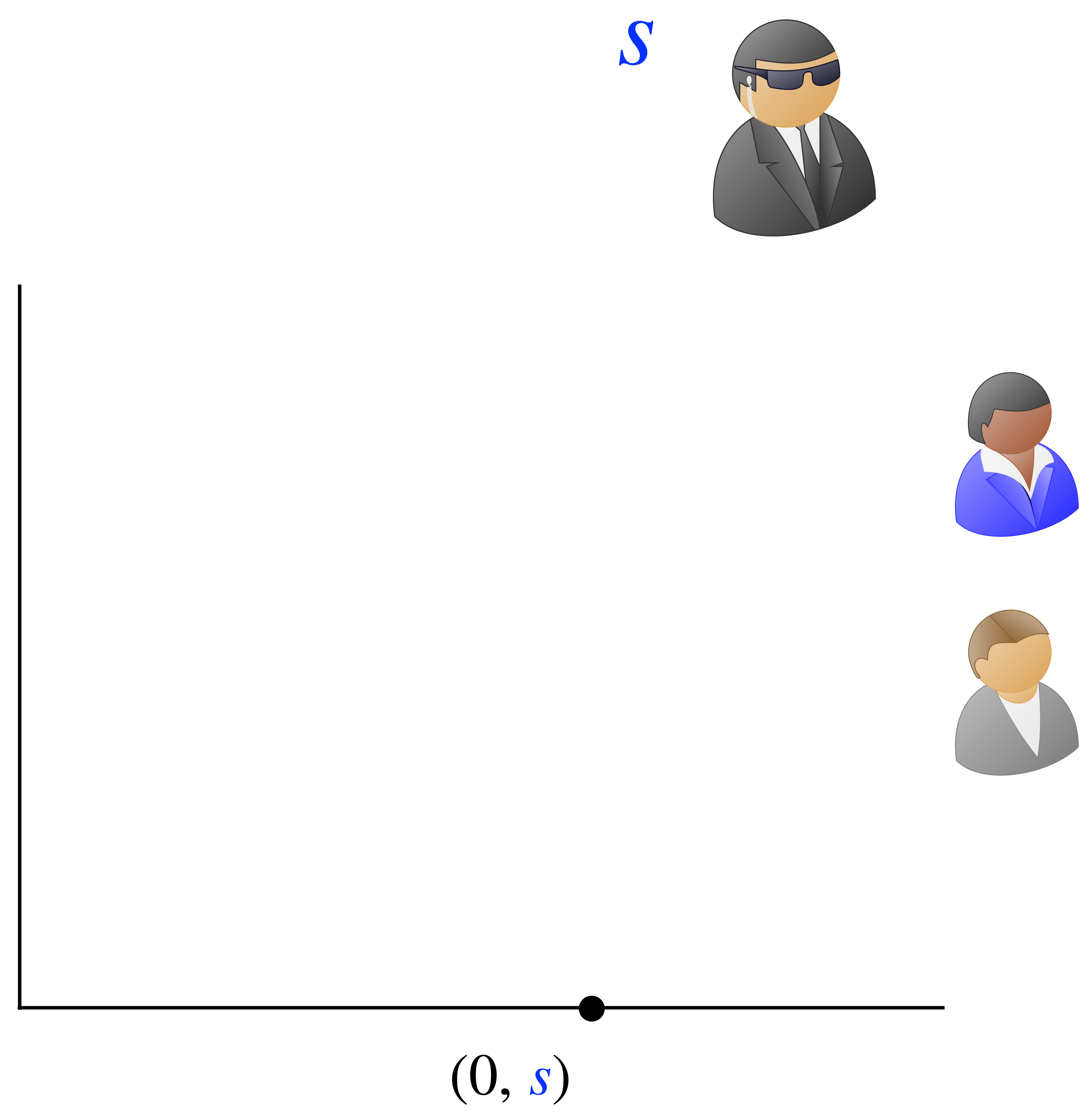
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party

# Shamir Secret Sharing

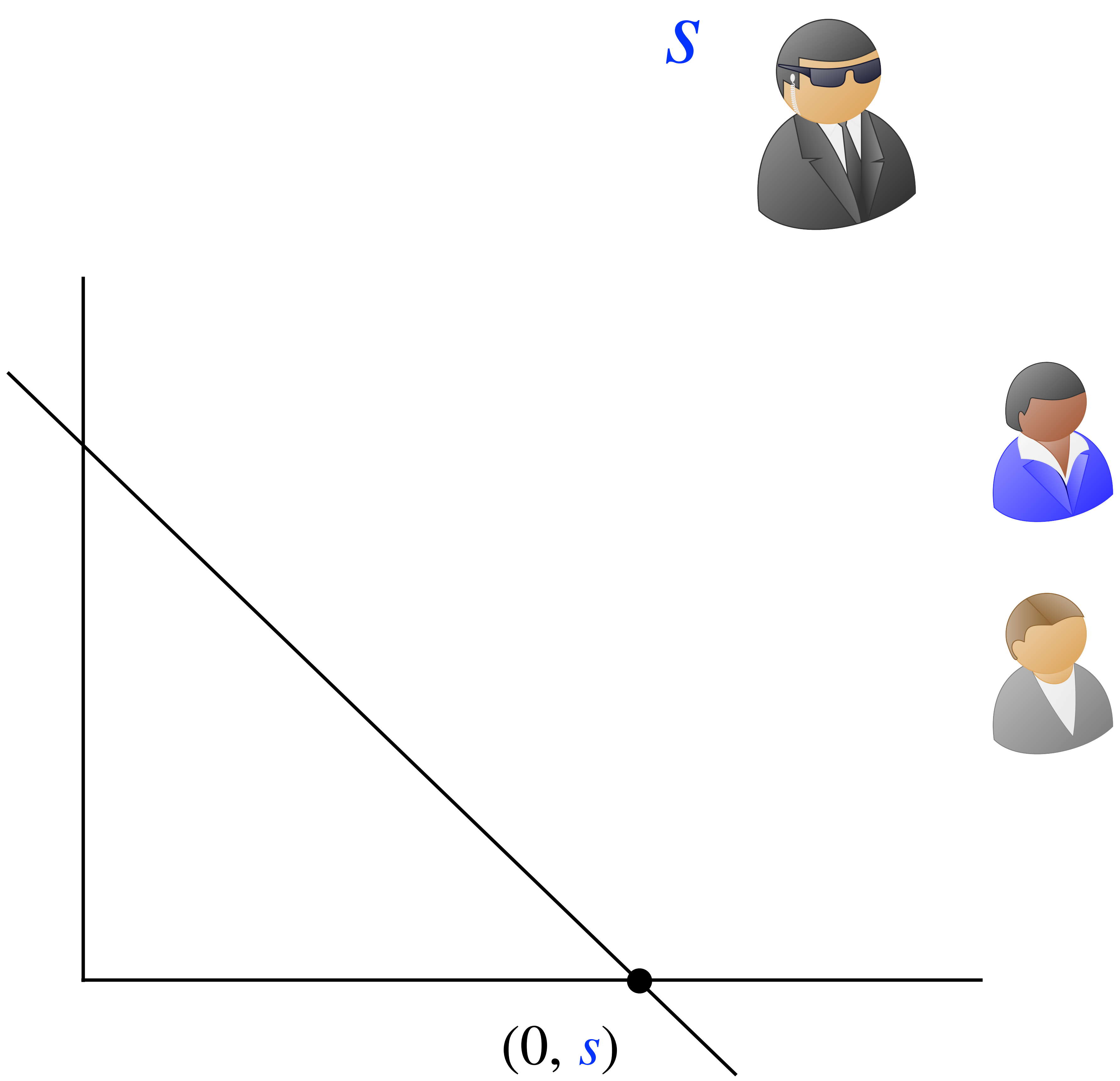
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party

# Shamir Secret Sharing

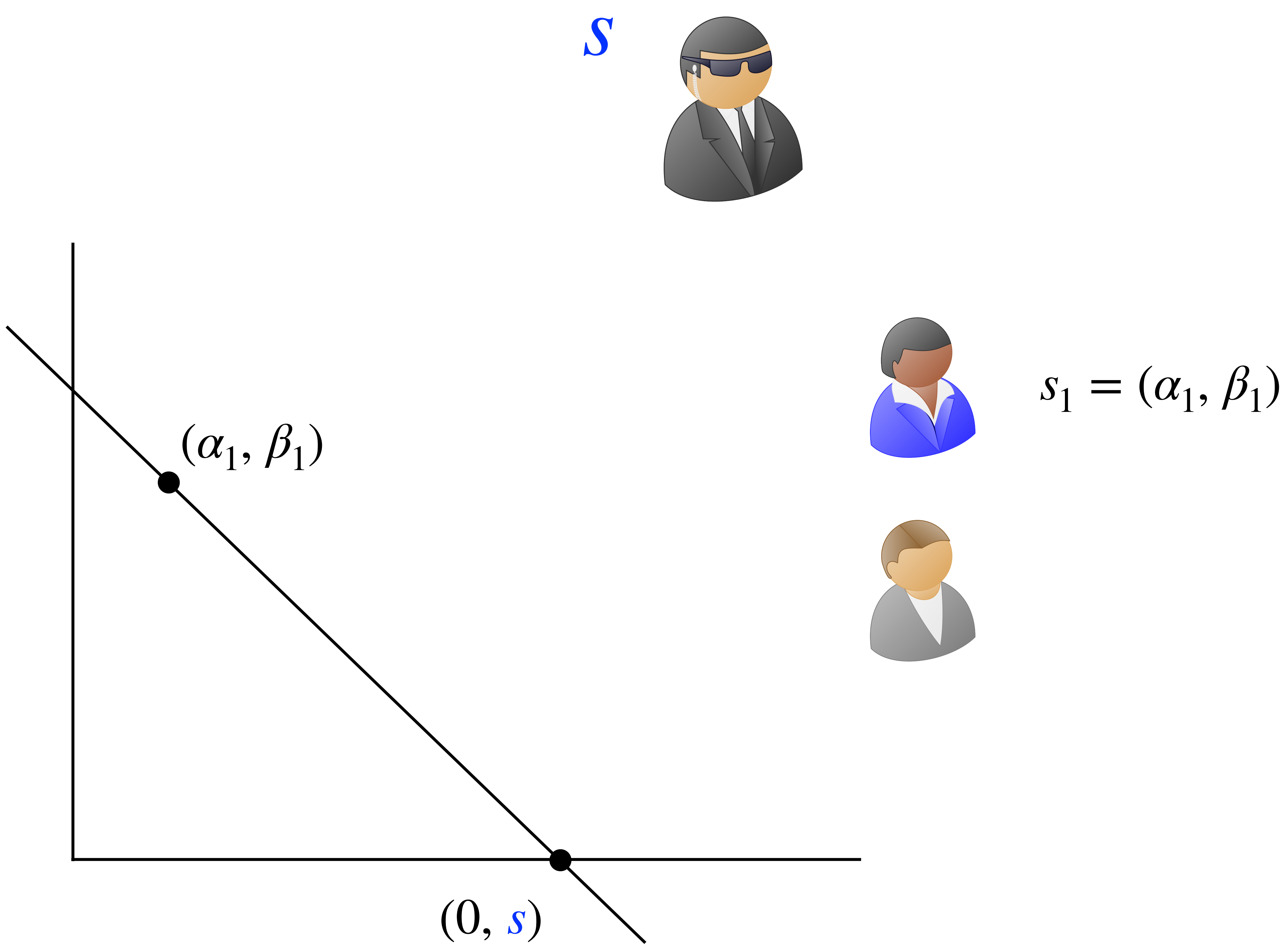
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party

# Shamir Secret Sharing

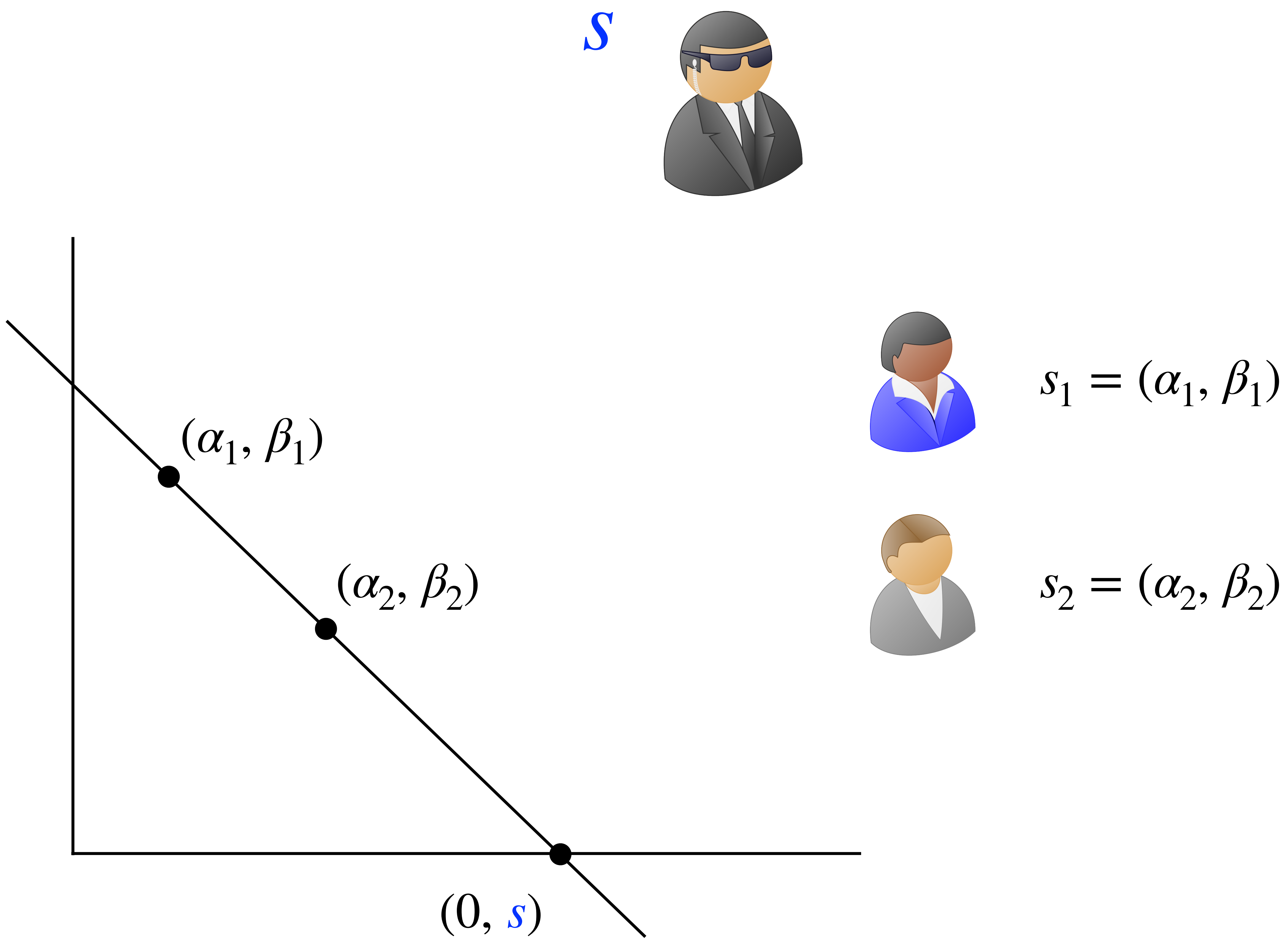
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party

# Shamir Secret Sharing

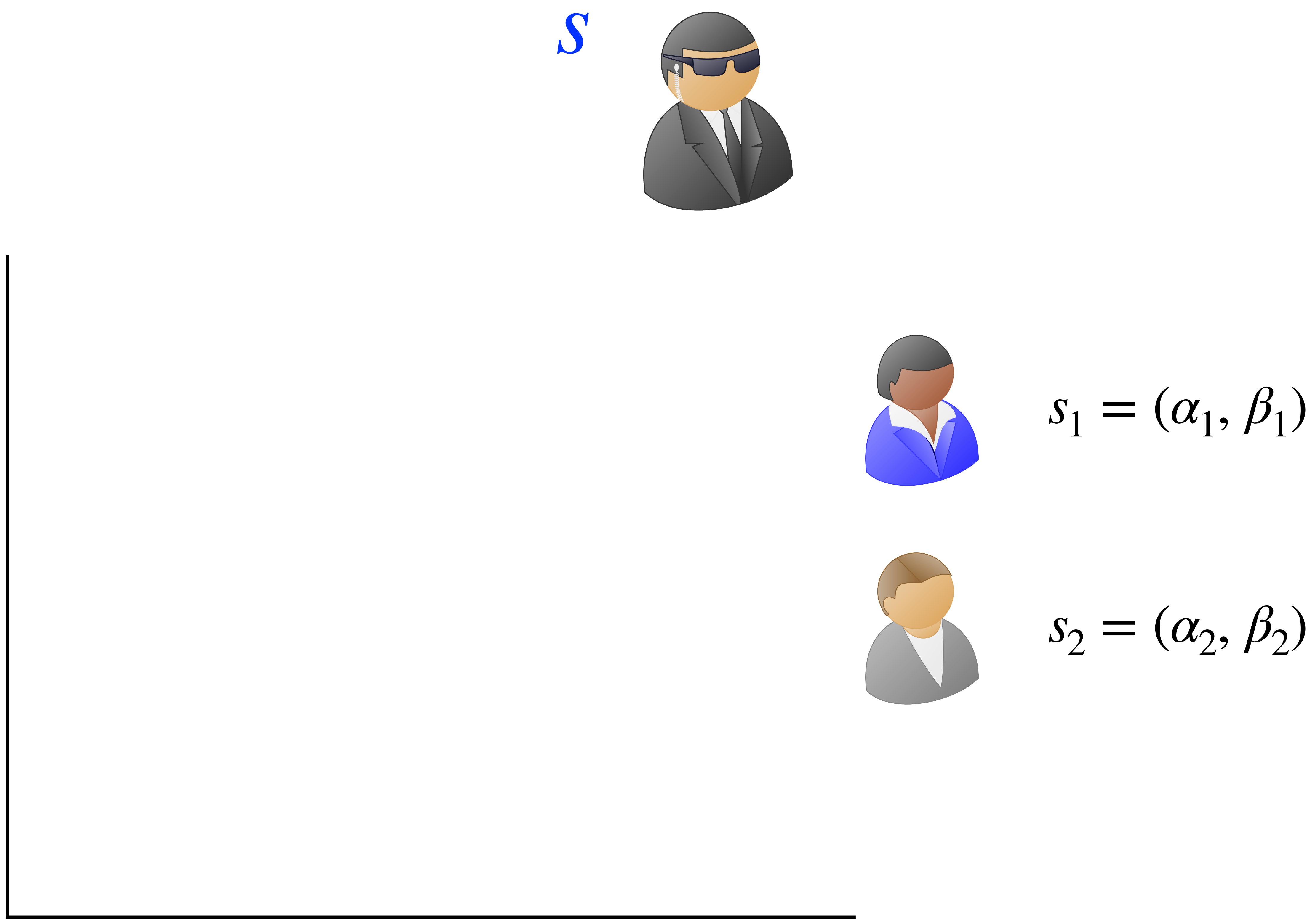
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party

# Shamir Secret Sharing

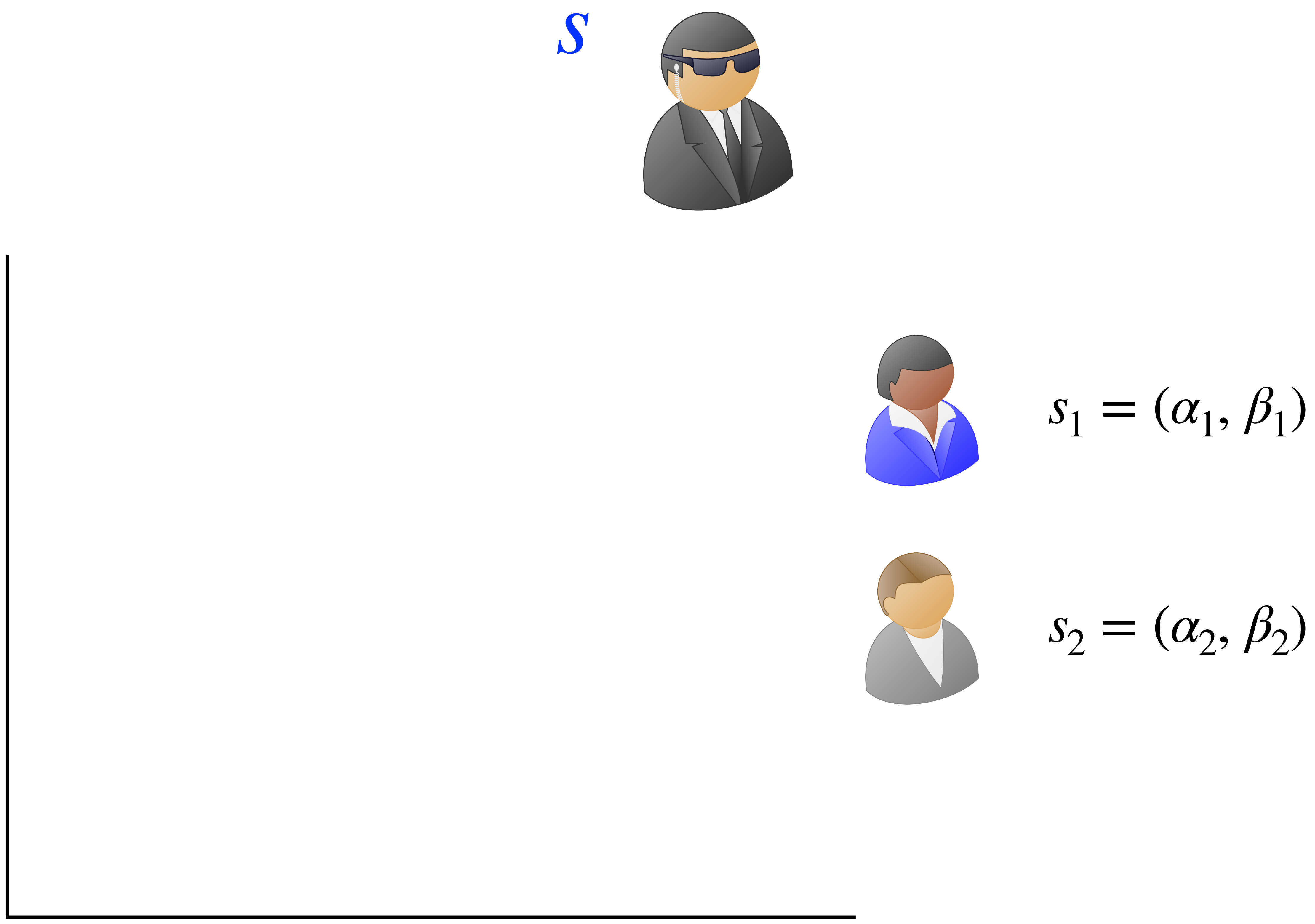
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party

# Shamir Secret Sharing

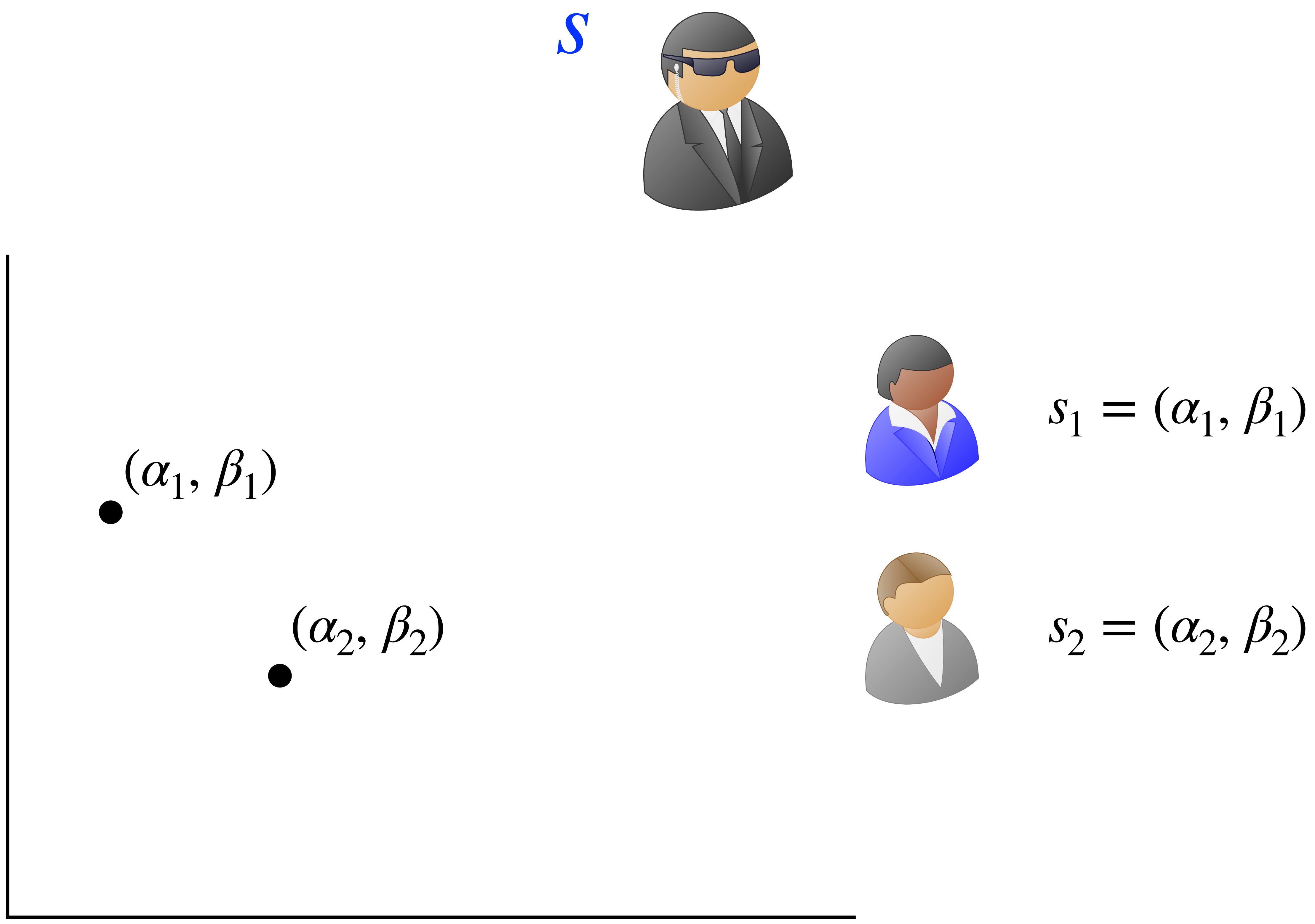
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line

# Shamir Secret Sharing

## 2-out-of-2 Sharing



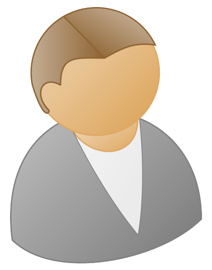

- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line

# Shamir Secret Sharing

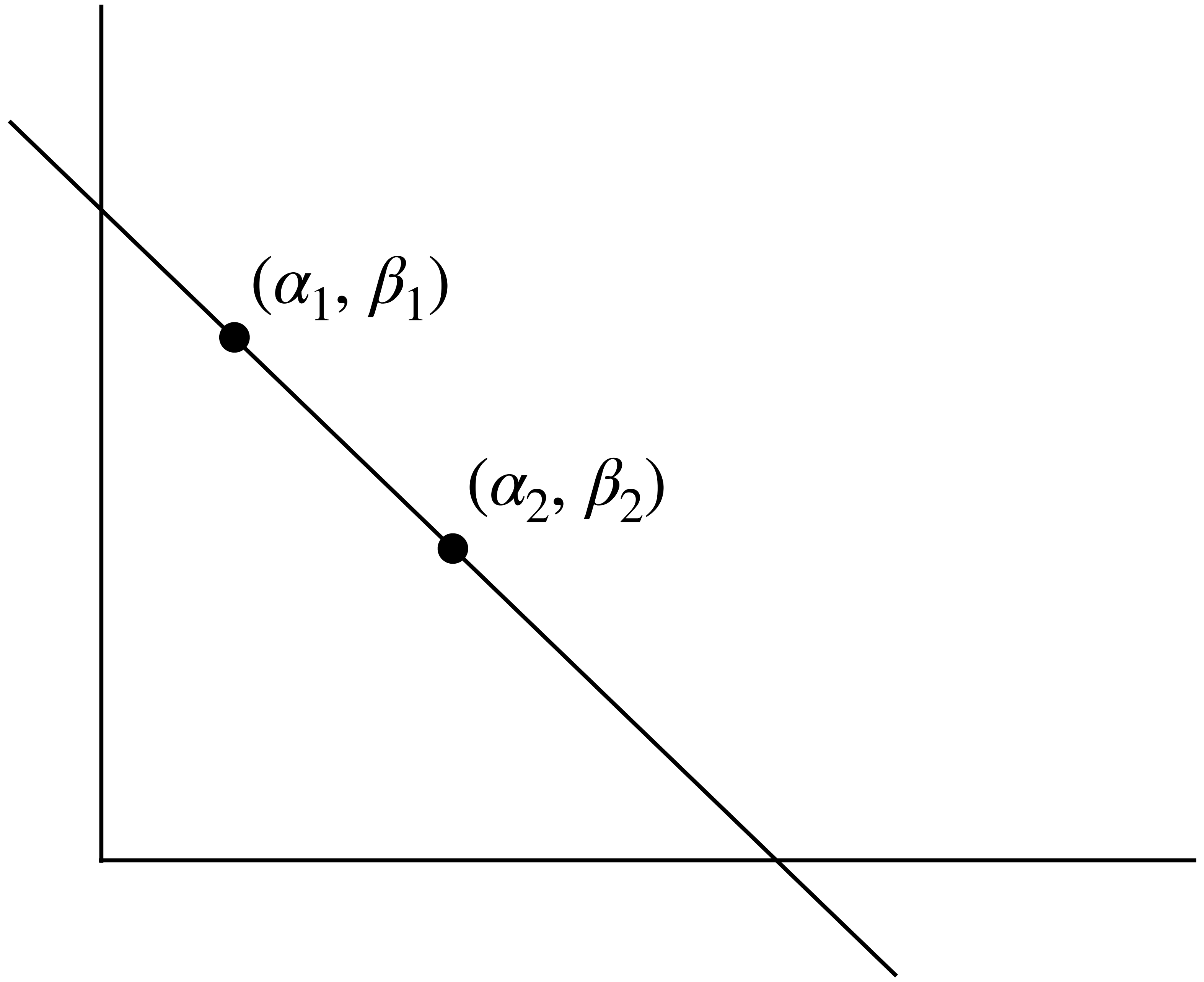
## 2-out-of-2 Sharing

$s$

$s_1 = (\alpha_1, \beta_1)$



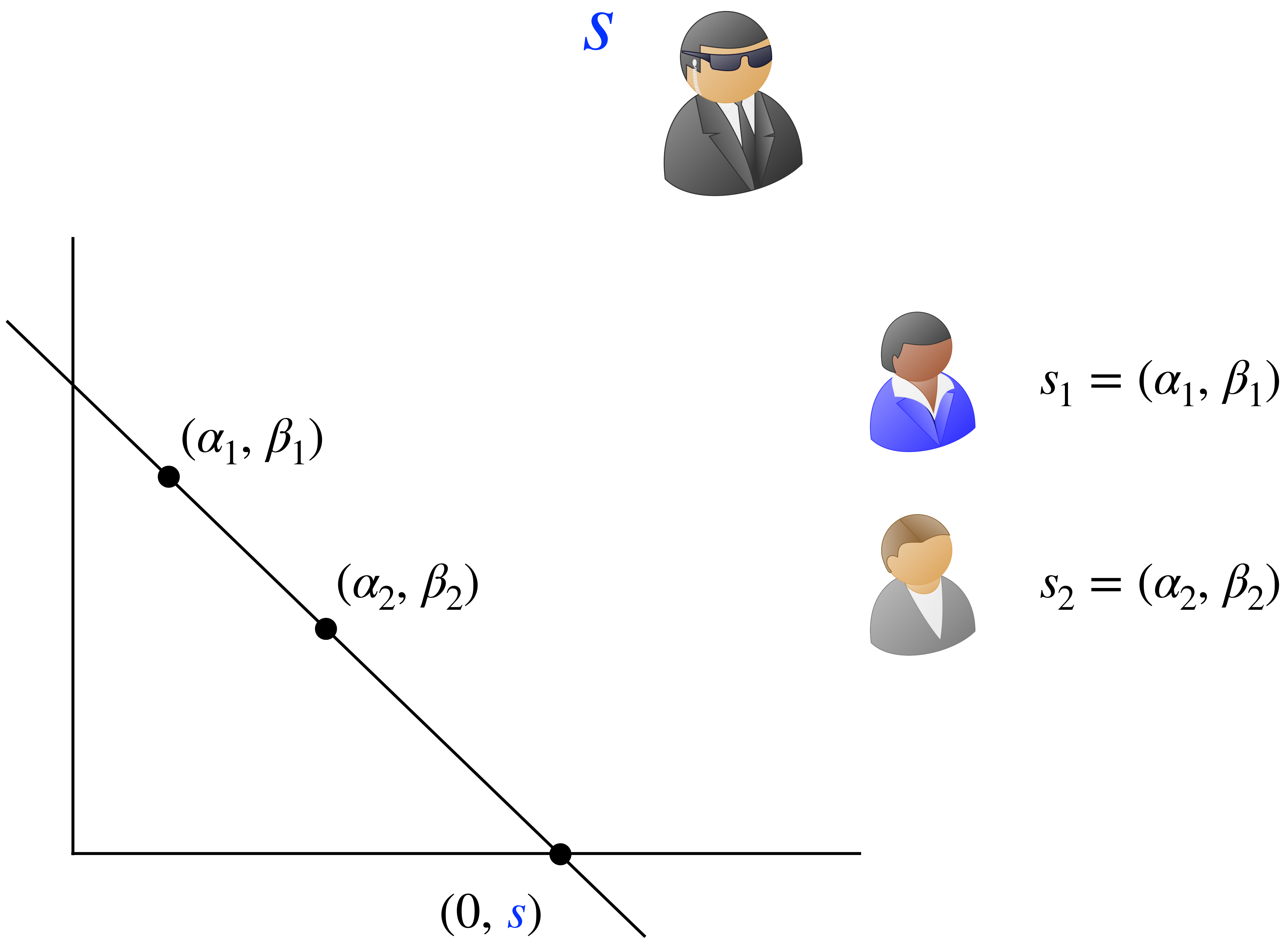
$s_2 = (\alpha_2, \beta_2)$




- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line

# Shamir Secret Sharing

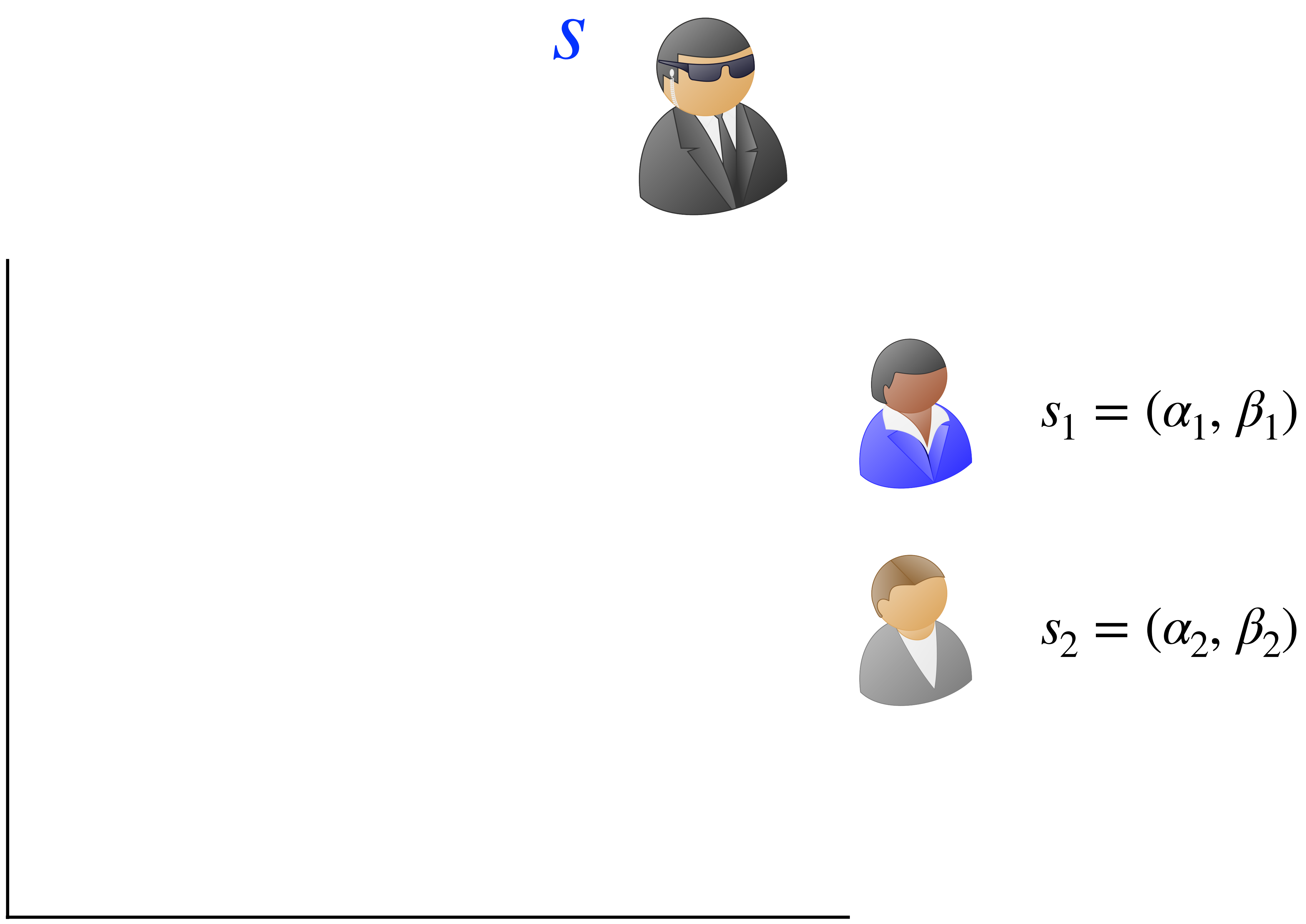
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line

# Shamir Secret Sharing

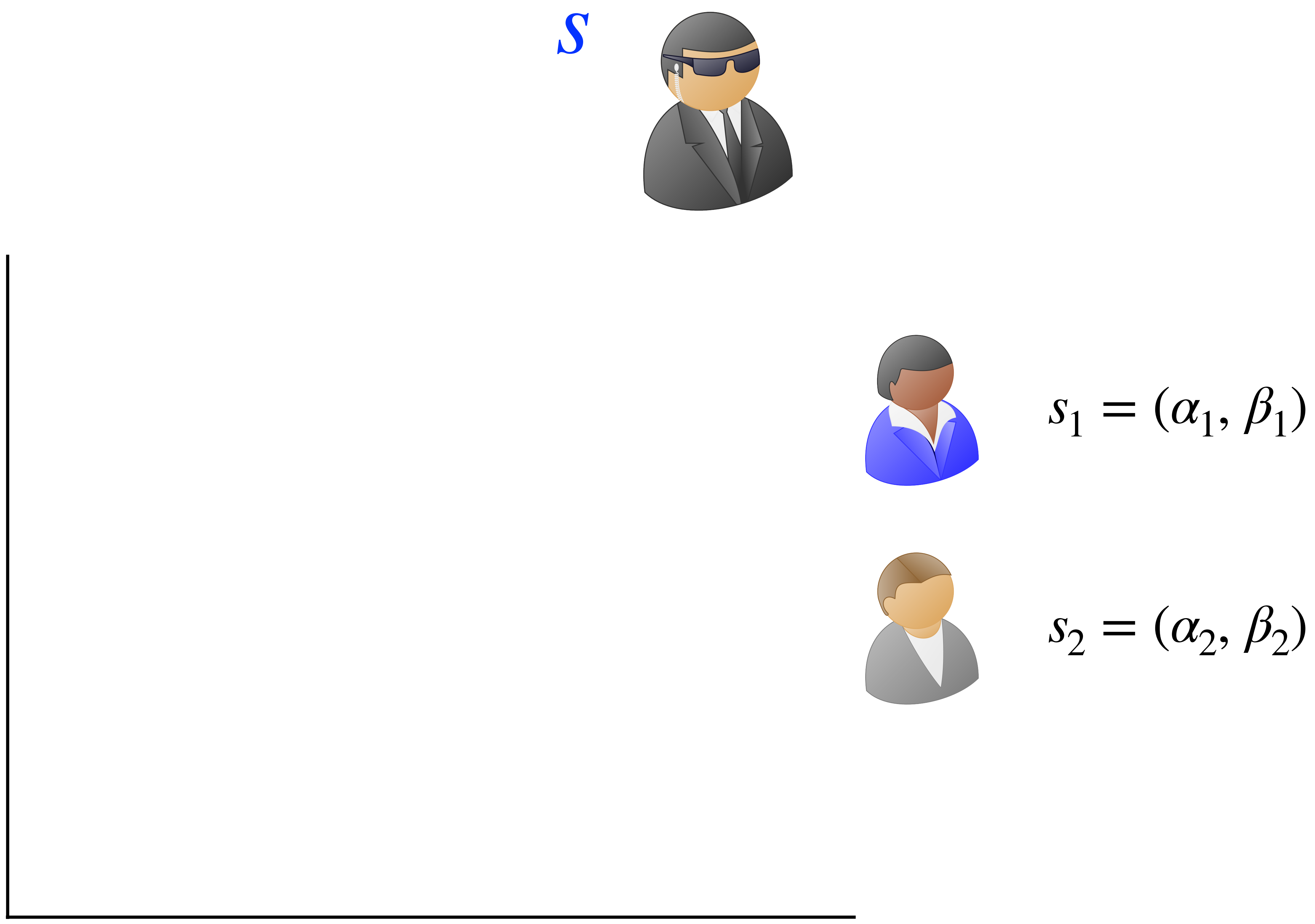
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line

# Shamir Secret Sharing

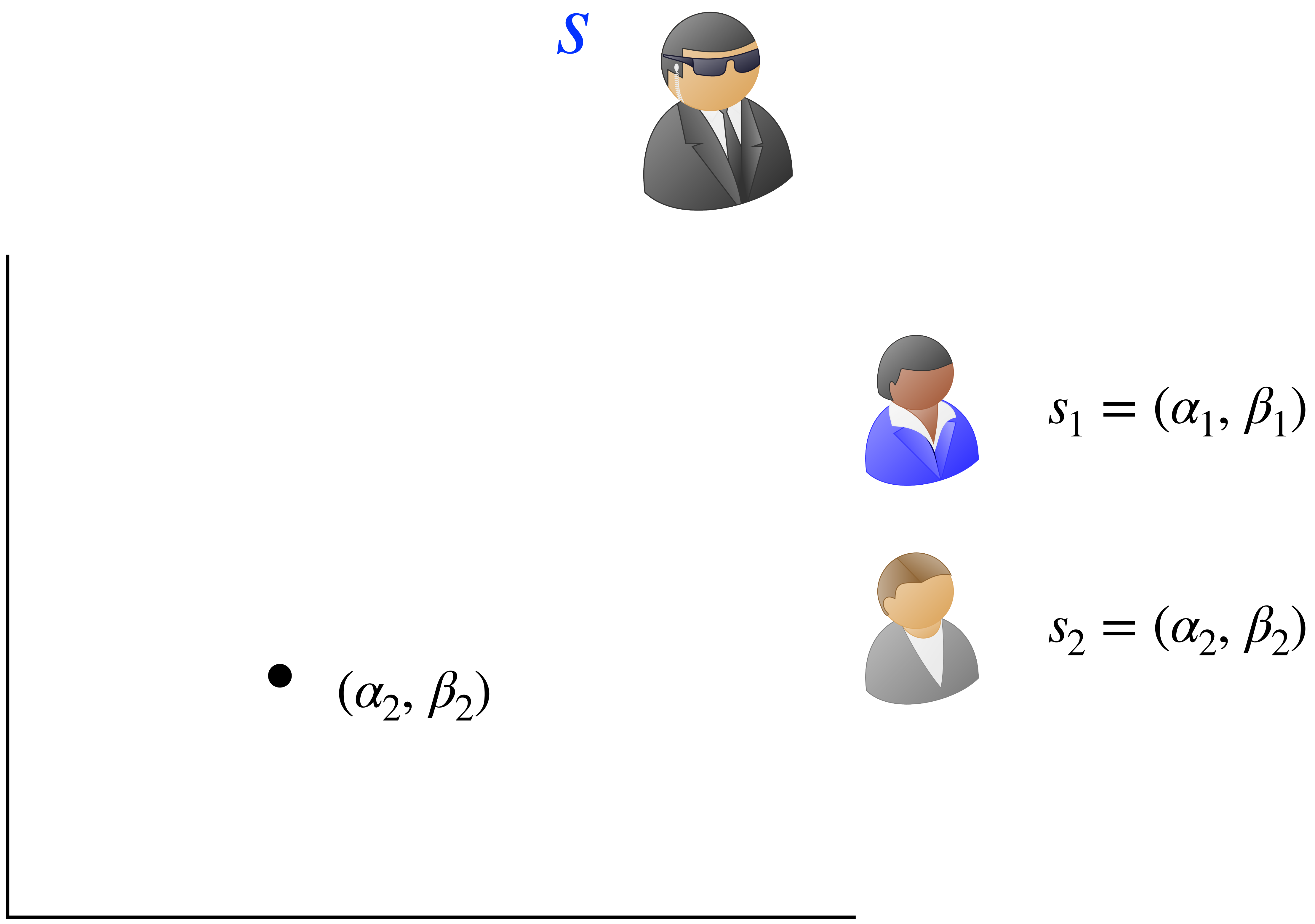
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

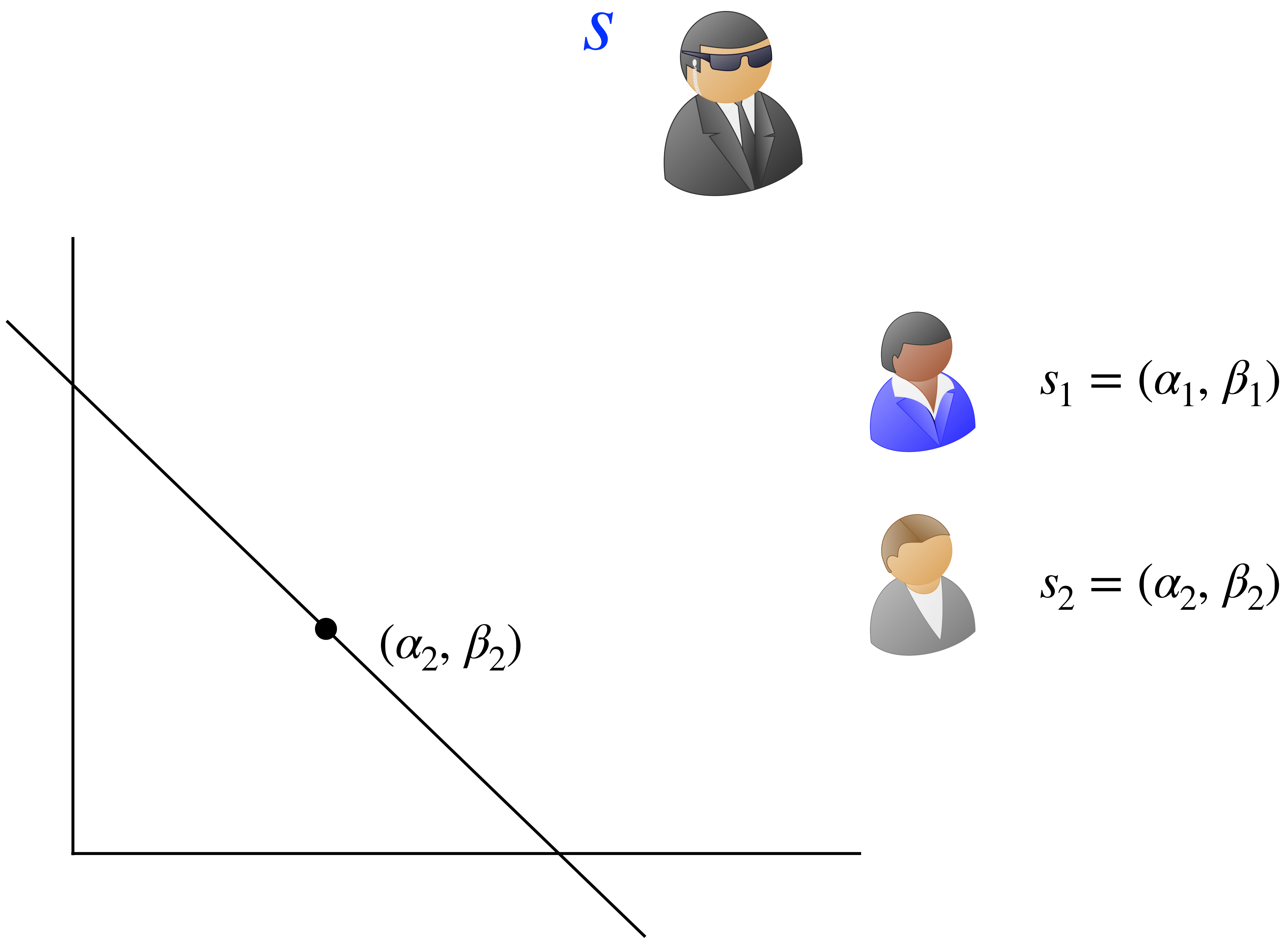
## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing

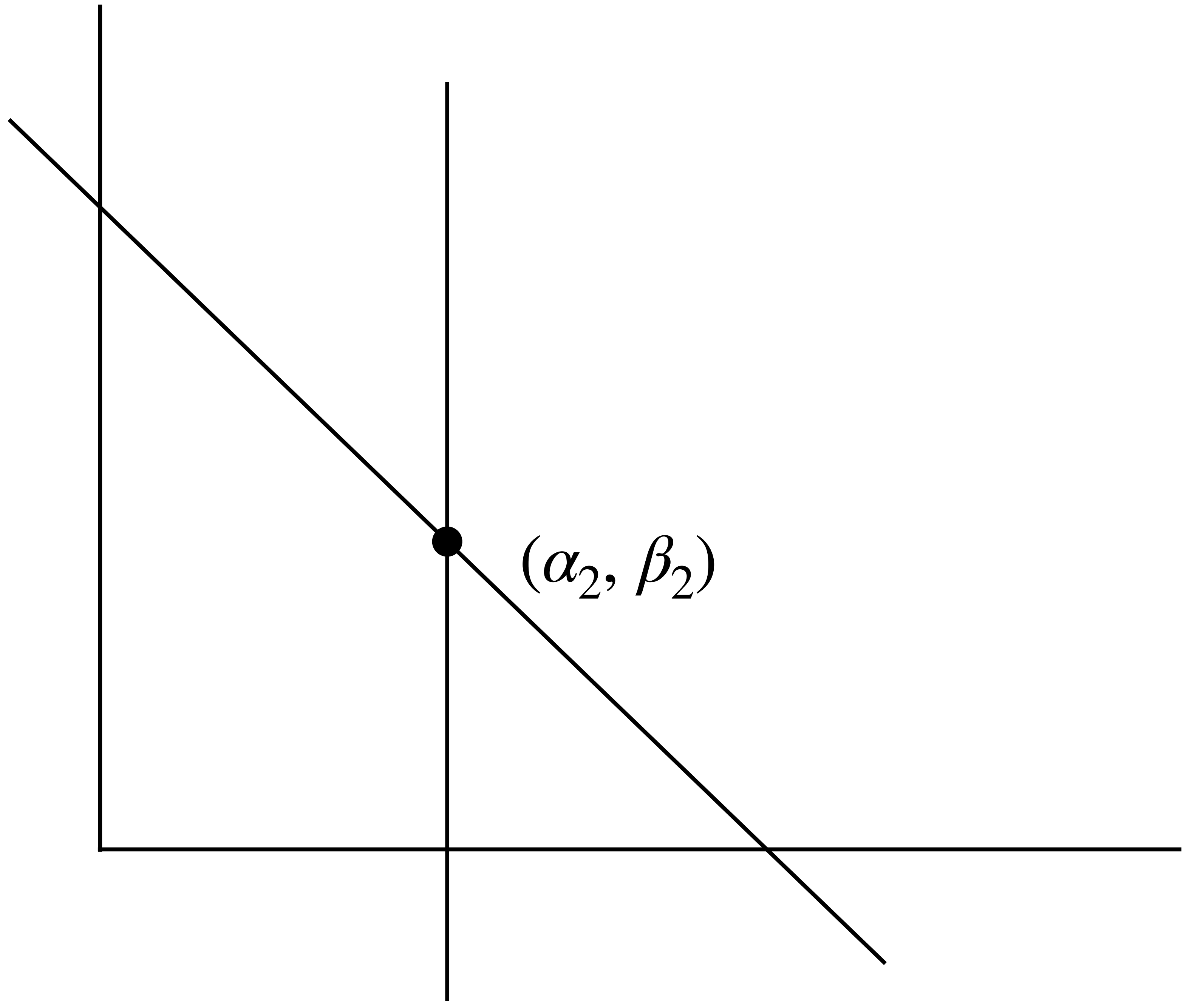


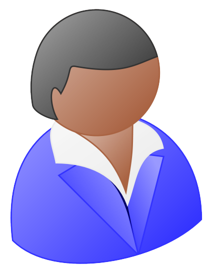

- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

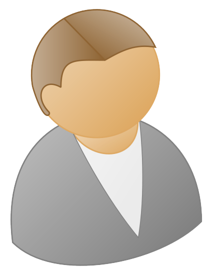
## 2-out-of-2 Sharing

$s$



$s_1 = (\alpha_1, \beta_1)$

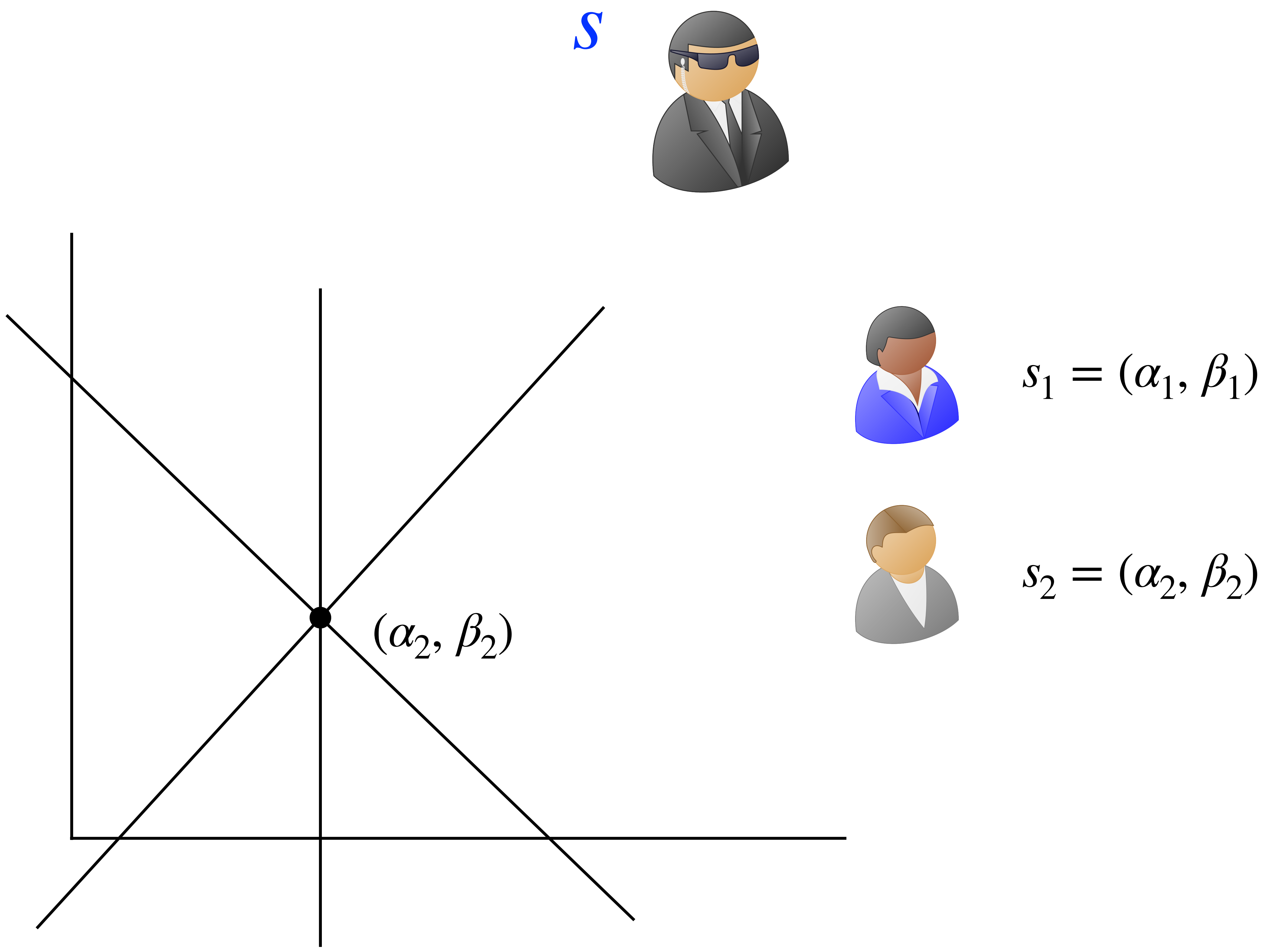


$s_2 = (\alpha_2, \beta_2)$

- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing

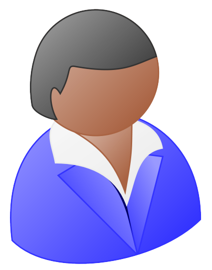



- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

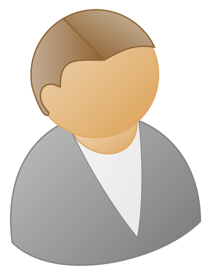
# Shamir Secret Sharing

## 2-out-of-2 Sharing

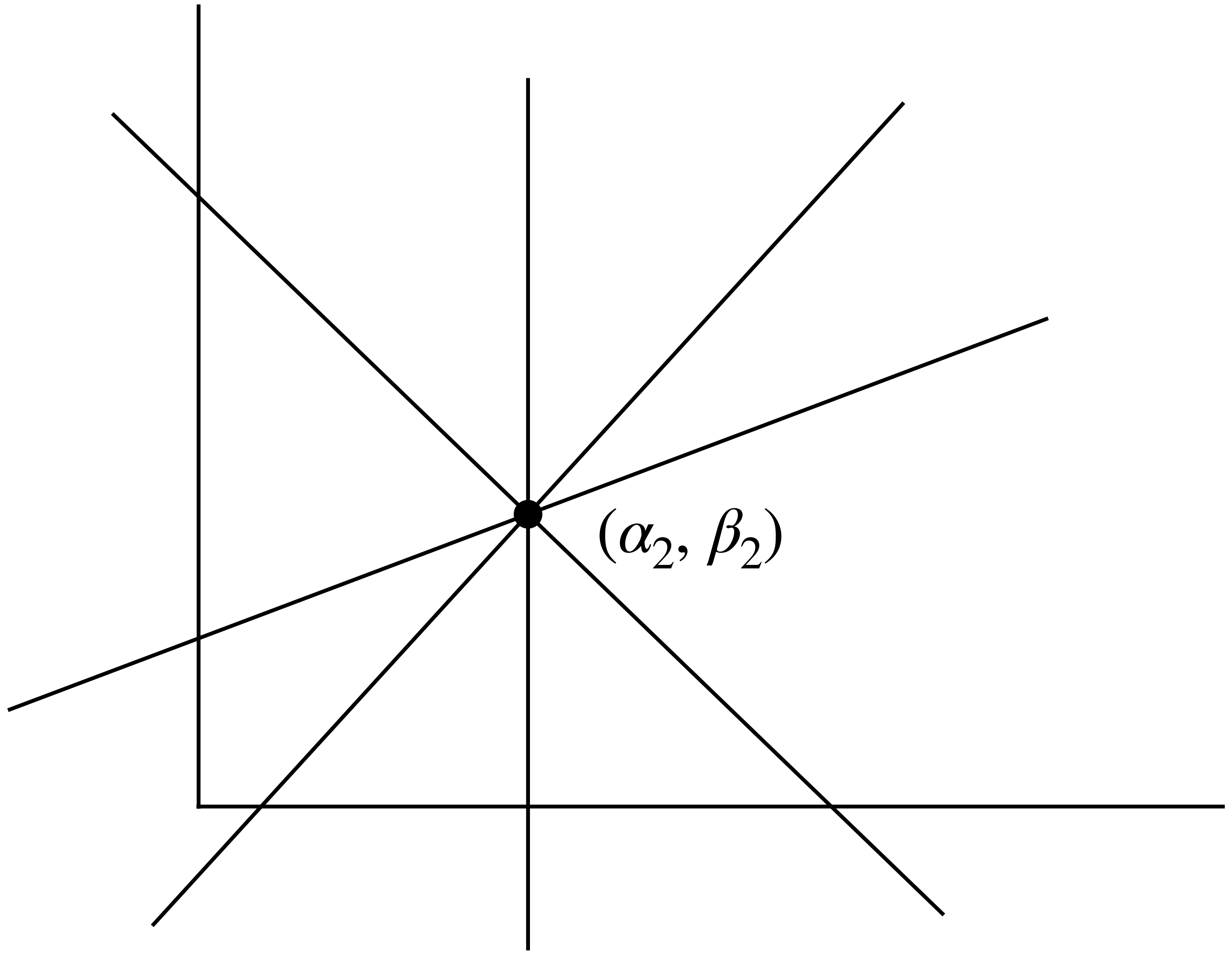
$s$



$s_1 = (\alpha_1, \beta_1)$



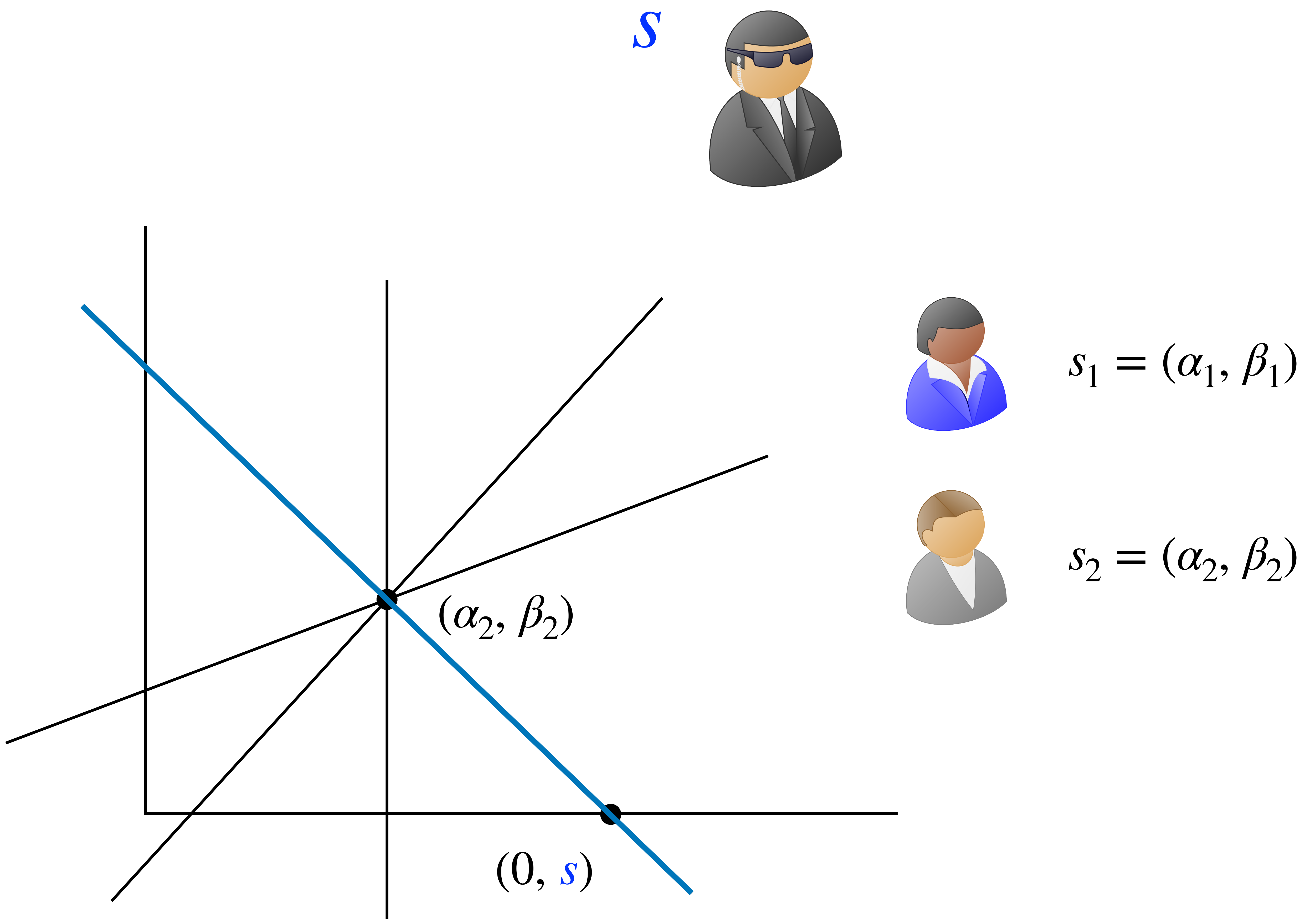
$s_2 = (\alpha_2, \beta_2)$




- Share algorithm (intuition)
    - Choose a “random line” that passes through $(0, s)$
    - Give one point to each party
- Correctness
    - 2 points uniquely define a line
- Privacy (intuition)
    - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing




- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing

- Share algorithm
  - Sample $c_1 \leftarrow \mathbb{Z}_p$ uniformly at random and set $p(x) = c_1 \cdot x \; + \; s$
  - Let $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ such that $\alpha_1 \neq 0$ and $\alpha_2 \neq 0$
  - Compute

$$s_1 = (\alpha_1, \beta_1 = p(\alpha_1)), \text{ and}$$

$$s_2 = (\alpha_2, \beta_2 = p(\alpha_2))$$

- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing

- Share algorithm
  - Sample $c_1 \leftarrow \mathbb{Z}_p$ uniformly at random and set $p(x) = c_1 \cdot x \ + \ s$
  - Let $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ such that $\alpha_1 \neq 0$ and $\alpha_2 \neq 0$
  - Compute

$$s_1 = (\alpha_1, \beta_1 = p(\alpha_1)), \text{ and}$$

$$s_2 = (\alpha_2, \beta_2 = p(\alpha_2))$$

- Correctness
  - 2 points uniquely define a degree-1 polynomial over $\mathbb{Z}_p$

- Share algorithm (intuition)
  - Choose a “random line” that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing

- Share algorithm
  - Sample $c_1 \leftarrow \mathbb{Z}_p$ uniformly at random and set $p(x) = c_1 \cdot x \ + \ s$
  - Let $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ such that $\alpha_1 \neq 0$ and $\alpha_2 \neq 0$
  - Compute

$$s_1 = (\alpha_1, \beta_1 = p(\alpha_1)), \text{ and}$$

$$s_2 = (\alpha_2, \beta_2 = p(\alpha_2))$$

- Correctness
  - 2 points uniquely define a degree-1 polynomial over $\mathbb{Z}_p$
- Privacy
  - There are $p$ lines passing through any single point

- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## 2-out-of-2 Sharing

- Share algorithm
  - Sample $c_1 \leftarrow \mathbb{Z}_p$ uniformly at random and set $p(x) = c_1 \cdot x + s$
  - Let $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ such that $\alpha_1 \neq 0$ and $\alpha_2 \neq 0$
  - Compute

$$s_1 = (\alpha_1, \beta_1 = p(\alpha_1)), \text{ and}$$

$$s_2 = (\alpha_2, \beta_2 = p(\alpha_2))$$

- Correctness
  - 2 points uniquely define a degree-1 polynomial over $\mathbb{Z}_p$
- Privacy
  - There are $p$ lines passing through any single point

Why $\mathbb{Z}_p$?

- Share algorithm (intuition)
  - Choose a "random line" that passes through $(0, s)$
  - Give one point to each party
- Correctness
  - 2 points uniquely define a line
- Privacy (intuition)
  - Infinitely many lines through a given point

# Shamir Secret Sharing

## $k$-out-of-$n$ Secret Sharing

# Shamir Secret Sharing

## $k$-out-of-$n$ Secret Sharing

- **Share algorithm:** Let $s \in \mathbb{Z}_p$ be the secret and let $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p$ be distinct *non-zero* values
  - Sample $c_{k-1}, \ldots, c_1 \leftarrow \mathbb{Z}_p$ uniformly at random
  - Set $p(x) = s + \sum_{i=1}^{k-1} c_i \cdot x^i$
  - The $i$-th party's share is $s_i = (\alpha_i , \beta_i = p(\alpha_i))$

# Shamir Secret Sharing

## $k$-out-of-$n$ Secret Sharing

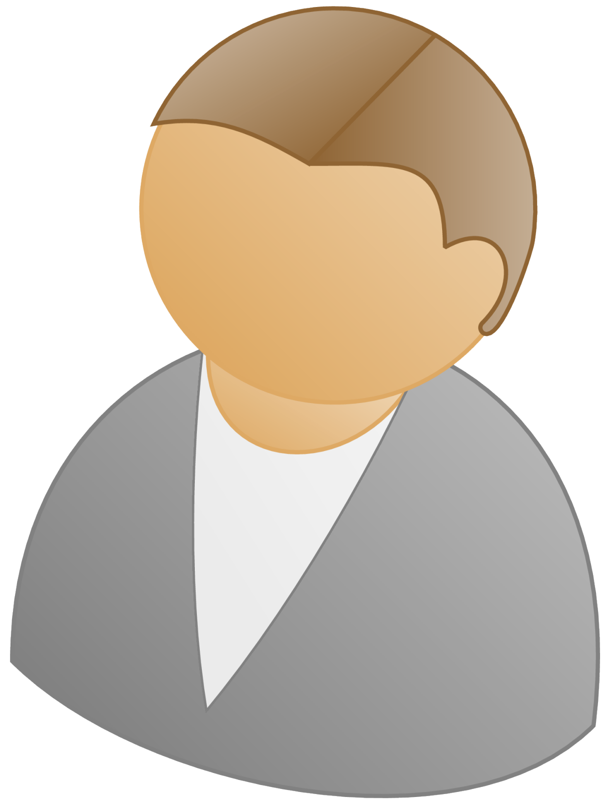
- **Share algorithm:** Let $s \in \mathbb{Z}_p$ be the secret and let $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p$ be distinct *non-zero* values
  - Sample $c_{k-1}, \ldots, c_1 \leftarrow \mathbb{Z}_p$ uniformly at random
  - Set $p(x) = s + \sum_{i=1}^{k-1} c_i \cdot x^i$
  - The $i$-th party's share is $s_i = (\alpha_i , \beta_i = p(\alpha_i))$
- **Correctness:** Any $k$ shares uniquely define a degree-$(k-1)$ polynomial $p(x)$. Compute the secret as $p(0)$.

# Shamir Secret Sharing

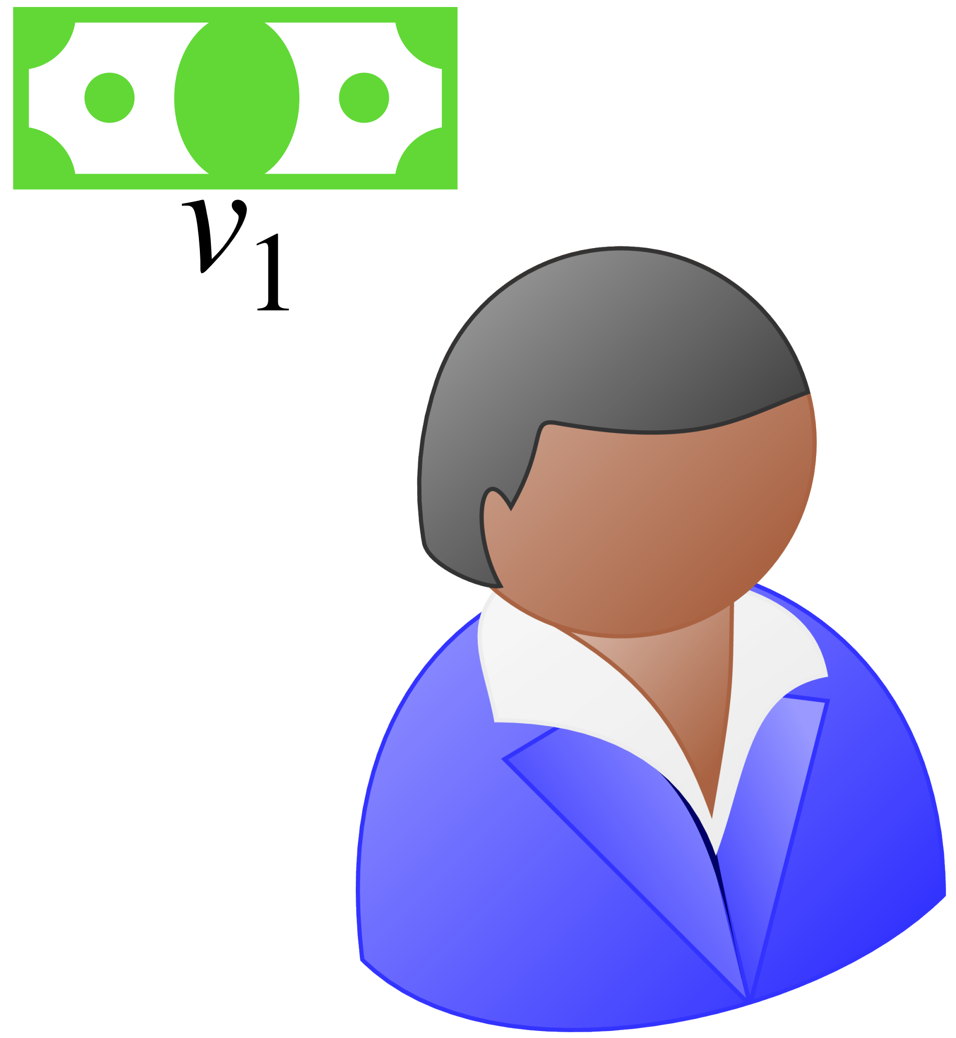
## $k$-out-of-$n$ Secret Sharing

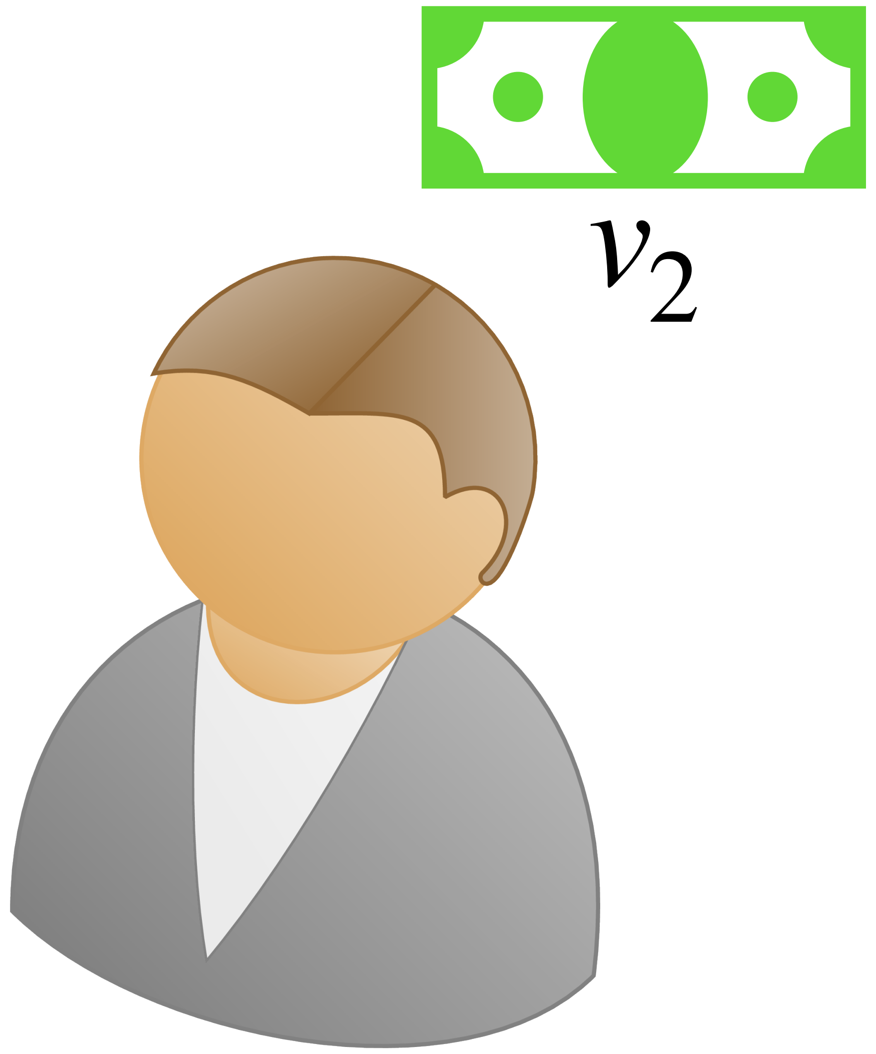
- **Share algorithm:** Let $s \in \mathbb{Z}_p$ be the secret and let $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p$ be distinct *non-zero* values
  - Sample $c_{k-1}, \ldots, c_1 \leftarrow \mathbb{Z}_p$ uniformly at random
  - Set $p(x) = s + \sum_{i=1}^{k-1} c_i \cdot x^i$
  - The $i$-th party's share is $s_i = (\alpha_i \, , \beta_i = p(\alpha_i))$
- **Correctness:** Any $k$ shares uniquely define a degree-$(k-1)$ polynomial $p(x)$. Compute the secret as $p(0)$.
- **Privacy:** For any subset of $k-1$ shares, there are $p$ degree-$(k-1)$ polynomials through them, any one of them could correspond to the secret.

# Multi-Party Computation: Motivation

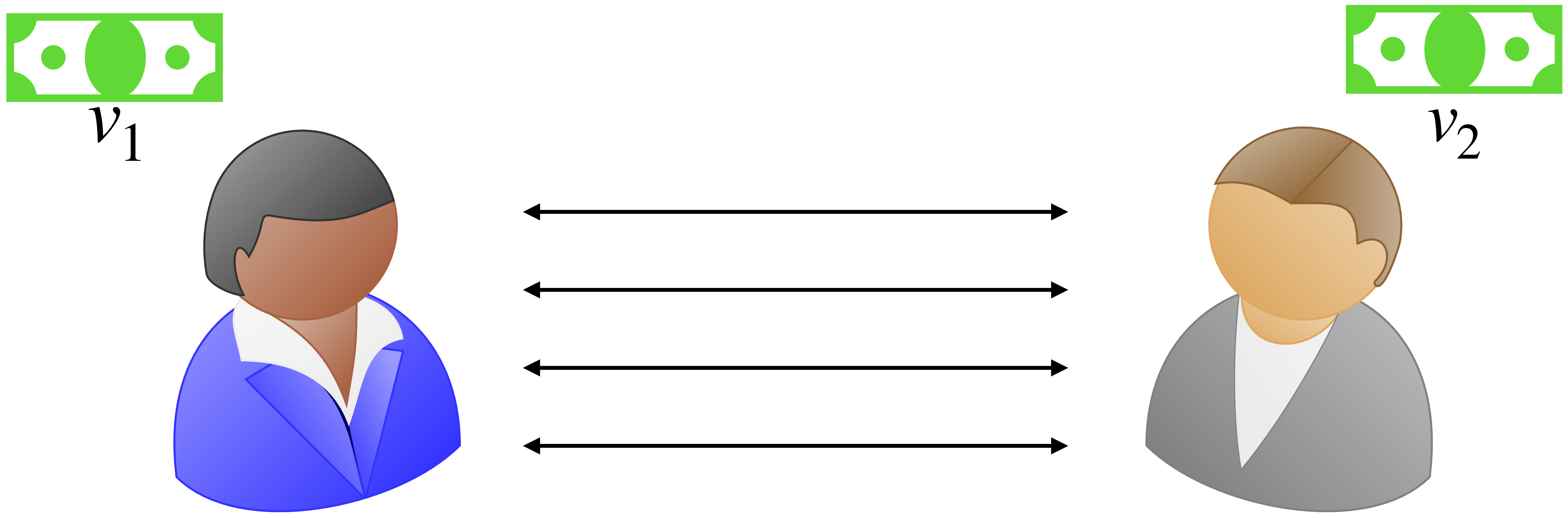


# Multi-Party Computation: Motivation

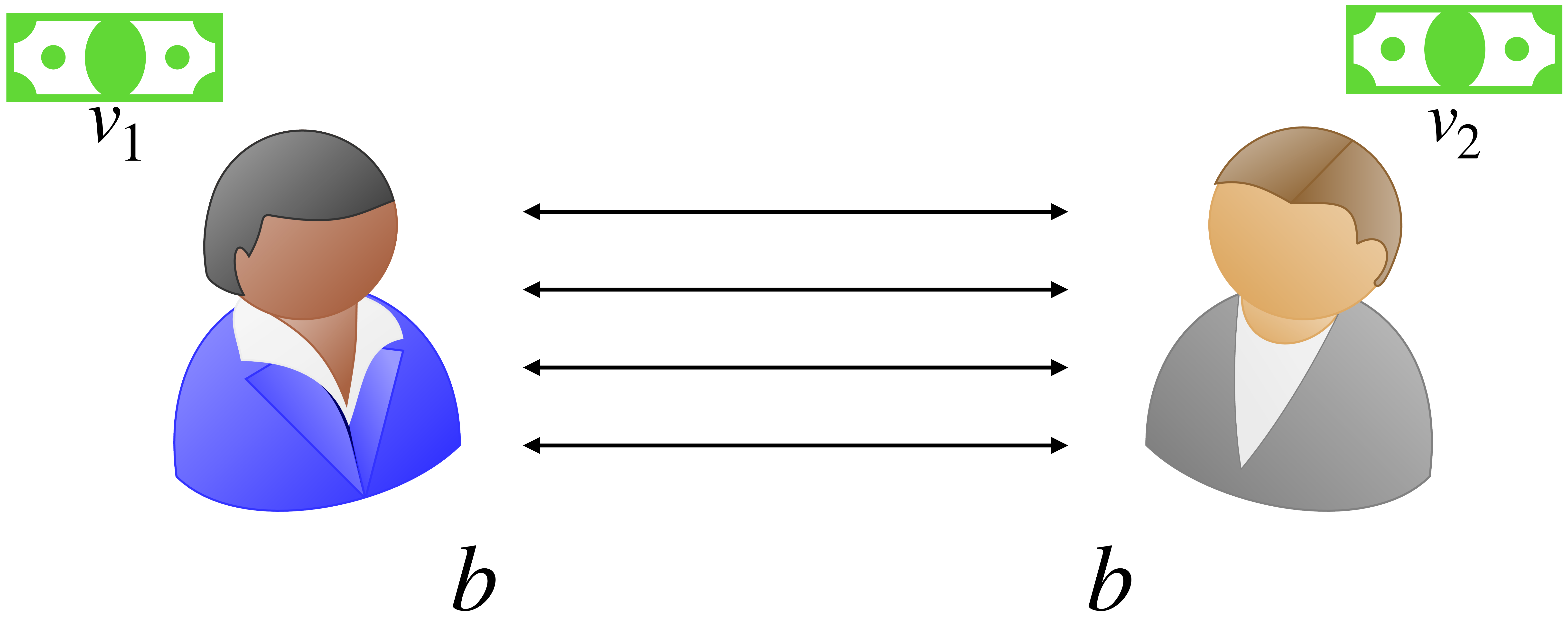


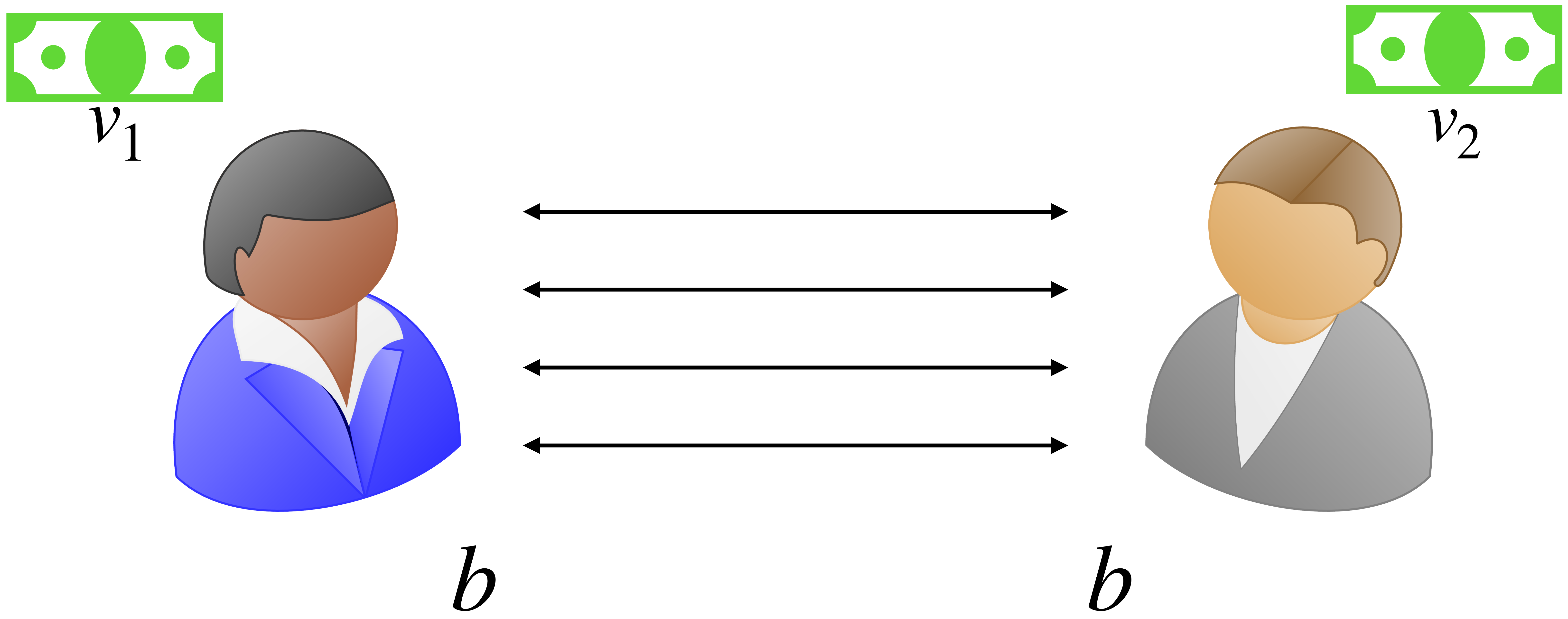
# Multi-Party Computation: Motivation



# Multi-Party Computation: Motivation

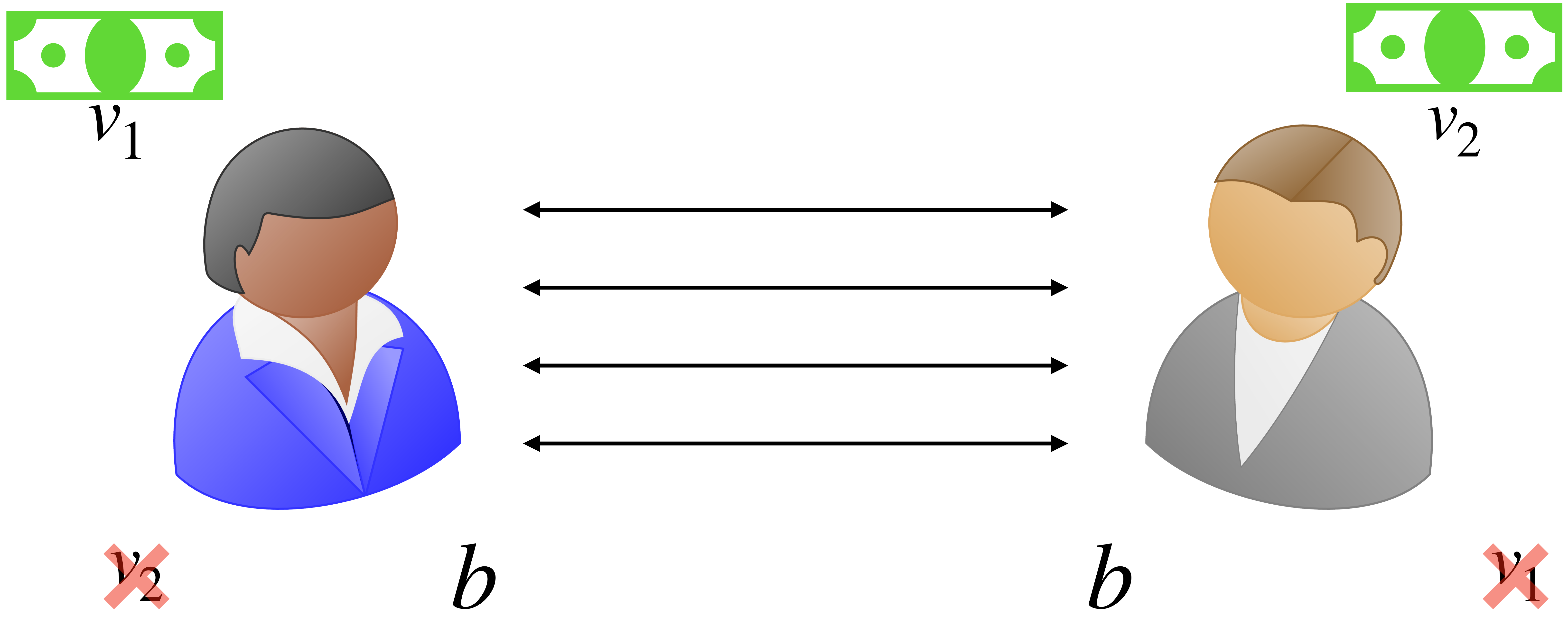


# Multi-Party Computation: Motivation




$b = 0$ if $v_1 < v_2$ else $b = 1$

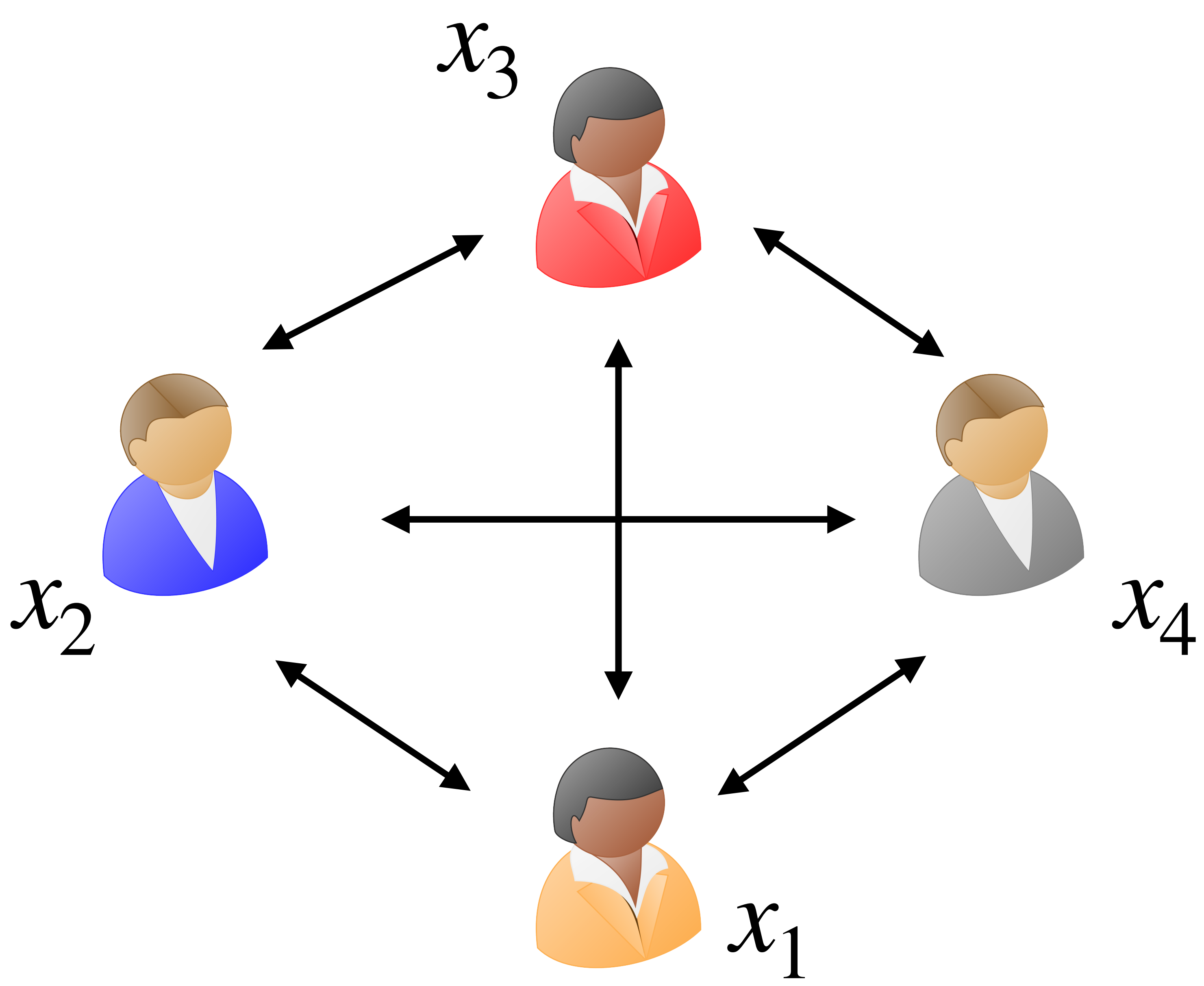
# Multi-Party Computation: Motivation




$$b = 0 \text{ if } v_1 < v_2 \text{ else } b = 1$$

Parties only learn the output, but nothing else about the other party's input

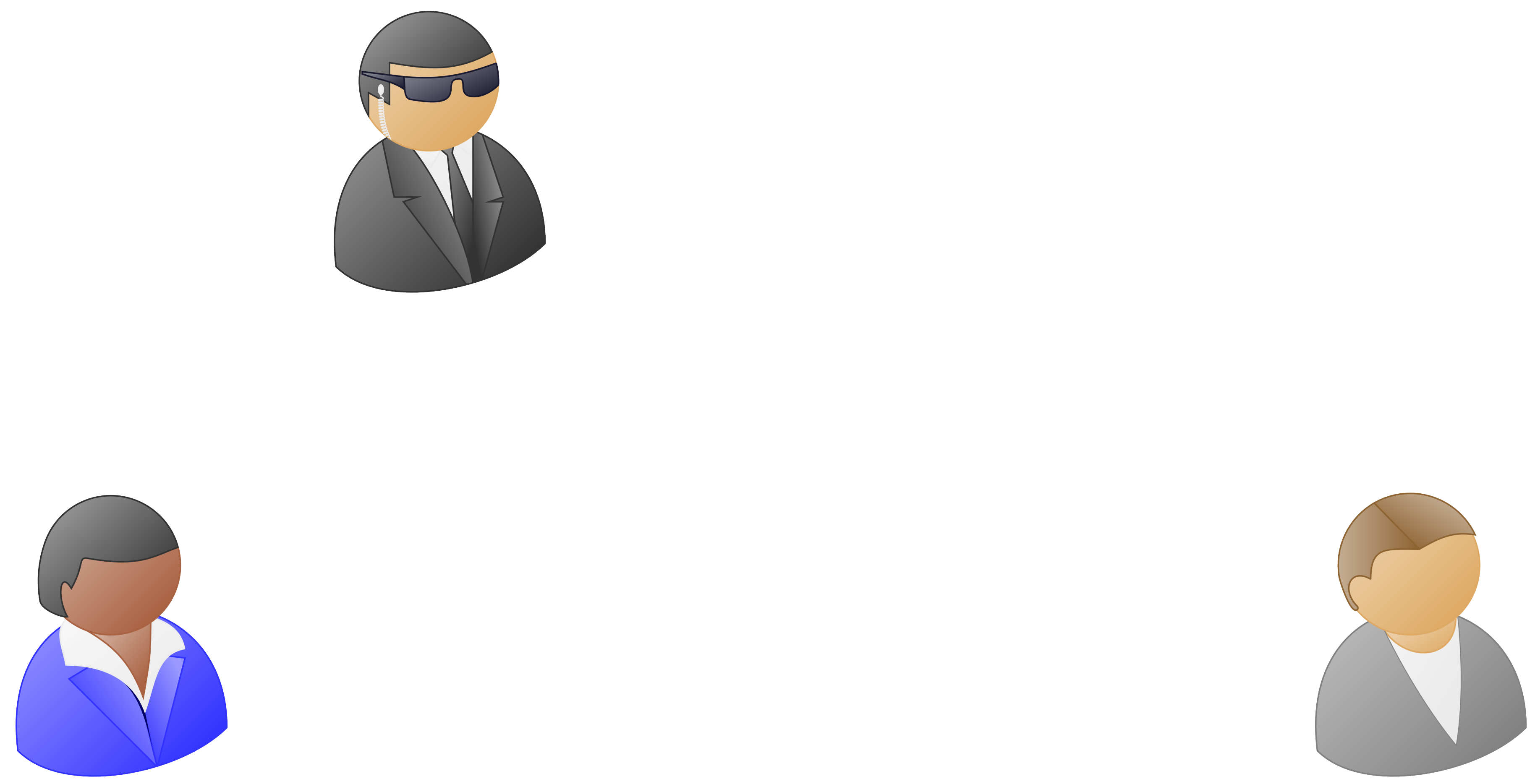
# Multi-Party Computation: Definition




$y = f(x_1, x_2, x_3, x_4)$

- Each party has an input
- Parties talk to each other to compute the output of a program/function
- No party learns anything beyond the function output

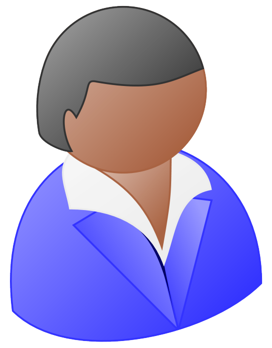
# Additively Homomorphic Secret Sharing

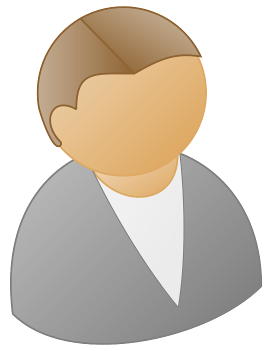


# Additively Homomorphic Secret Sharing

$u \in \{0, 1\}$

$v \in \{0, 1\}$





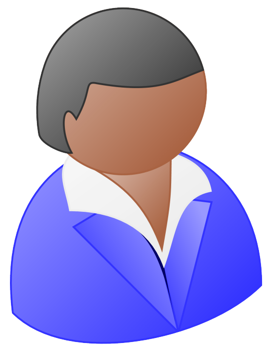
# Additively Homomorphic Secret Sharing

$u \in \{0, 1\}$

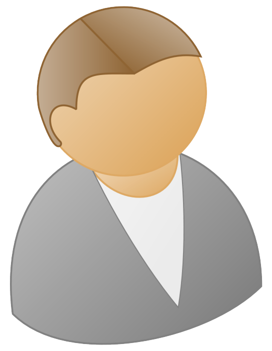
$v \in \{0, 1\}$

$u_1 \leftarrow \{0, 1\}, \quad u_2 = u \oplus u_1$

$v_1 \leftarrow \{0, 1\}, \quad v_2 = v \oplus v_1$



$u_1, v_1$

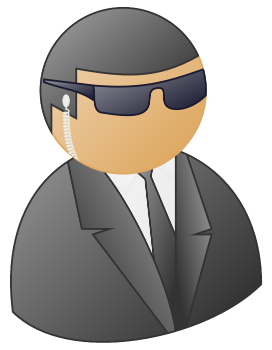


$u_2, v_2$

# Additively Homomorphic Secret Sharing

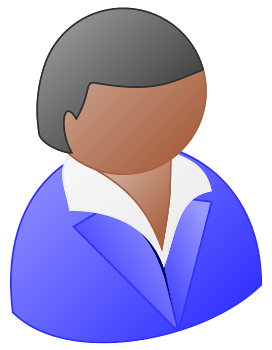
$\color{blue}{u} \in \{0, 1\}$

$\color{blue}{v} \in \{0, 1\}$



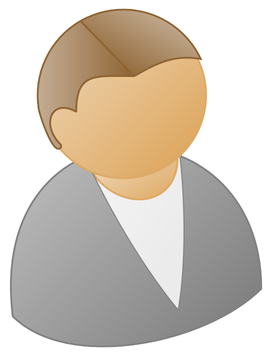
$$u_1 \leftarrow \{0, 1\}, \quad u_2 = \color{blue}{u} \color{black}{\oplus u_1}$$

$$v_1 \leftarrow \{0, 1\}, \quad v_2 = \color{blue}{v} \color{black}{\oplus v_1}$$



$u_1, v_1$

$w_1 = u_1 \oplus v_1$



$u_2, v_2$

$w_2 = u_2 \oplus v_2$

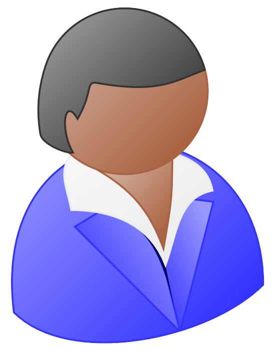
# Additively Homomorphic Secret Sharing

$u \in \{0, 1\}$

$v \in \{0, 1\}$

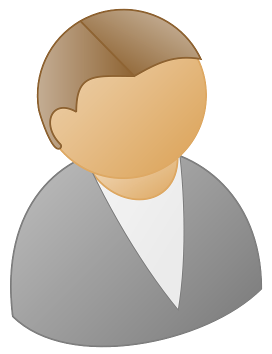
$u_1 \leftarrow \{0, 1\}, \quad u_2 = u \oplus u_1$

$v_1 \leftarrow \{0, 1\}, \quad v_2 = v \oplus v_1$



$u_1, v_1$

$w_1 = u_1 \oplus v_1$



$u_2, v_2$

$w_2 = u_2 \oplus v_2$

$$
\begin{aligned}
w_1 \oplus w_2 \\
&= u_1 \oplus v_1 \oplus u_2 \oplus v_2 \\
&= u_1 \oplus u_2 \oplus v_1 \oplus v_2 \quad = u \oplus v
\end{aligned}
$$

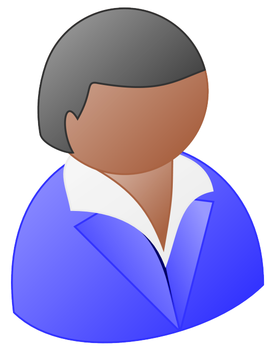
# Additively Homomorphic Secret Sharing

$u \in \{0,1\}$

$v \in \{0,1\}$

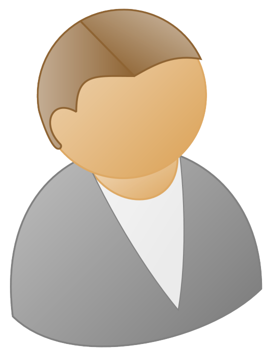
$u_1 \leftarrow \{0,1\}, \quad u_2 = u \oplus u_1$

$v_1 \leftarrow \{0,1\}, \quad v_2 = v \oplus v_1$



$u_1, v_1$

$w_1 = u_1 \oplus v_1$



$u_2, v_2$

$w_2 = u_2 \oplus v_2$

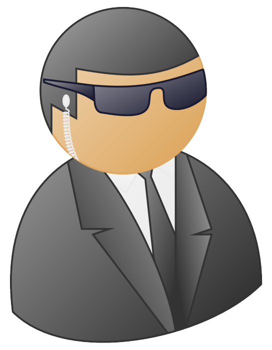
**Additive Homomorphism:** Each party can locally compute $w_1$ and $w_2$ which are shares of $u \oplus v$

$$
\begin{aligned}
w_1 \oplus w_2 \\
&= u_1 \oplus v_1 \oplus u_2 \oplus v_2 \\
&= u_1 \oplus u_2 \oplus v_1 \oplus v_2 \quad = u \oplus v
\end{aligned}
$$

# Additively Homomorphic Secret Sharing

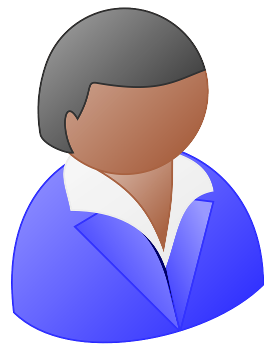
$u \in \{0, 1\}$

$v \in \{0, 1\}$



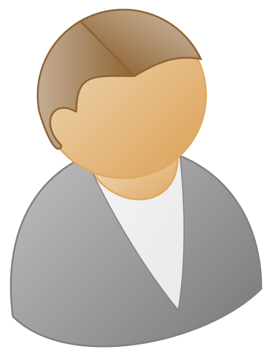
$u_1 \leftarrow \{0, 1\}, \quad u_2 = u \oplus u_1$

$v_1 \leftarrow \{0, 1\}, \quad v_2 = v \oplus v_1$



$u_1, v_1$

$w_1 = u_1 \oplus v_1$



$u_2, v_2$

$w_2 = u_2 \oplus v_2$

**Additive Homomorphism:** Each party can locally compute $w_1$ and $w_2$ which are shares of $u \oplus v$

Similarly, $k$-out-of-$n$ Shamir secret sharing supports addition over $\mathbb{Z}_p$

$$\begin{aligned} w_1 \oplus w_2 & \\ &= u_1 \oplus v_1 \oplus u_2 \oplus v_2 \\ &= u_1 \oplus u_2 \oplus v_1 \oplus v_2 \quad = u \oplus v \end{aligned}$$

# Multi-Party Computation

- We will compute **Boolean circuits** — these capture all computable program (though for real-life programs, the circuits might be impractically large!)
- Boolean circuits: AND gates, and XOR gates
- Additive homomorphism can help evaluate XOR gates
- How do we evaluate AND gates?