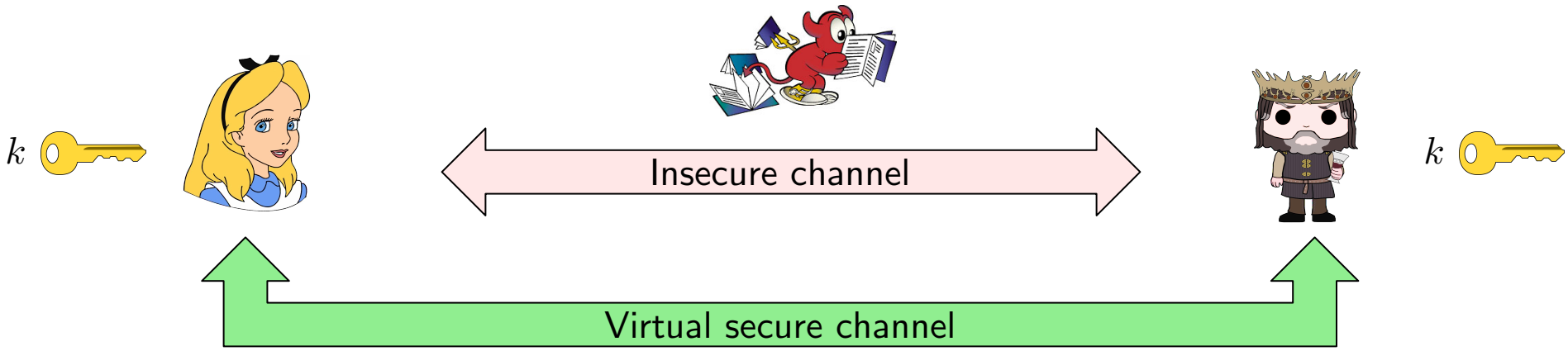
# Key Distribution

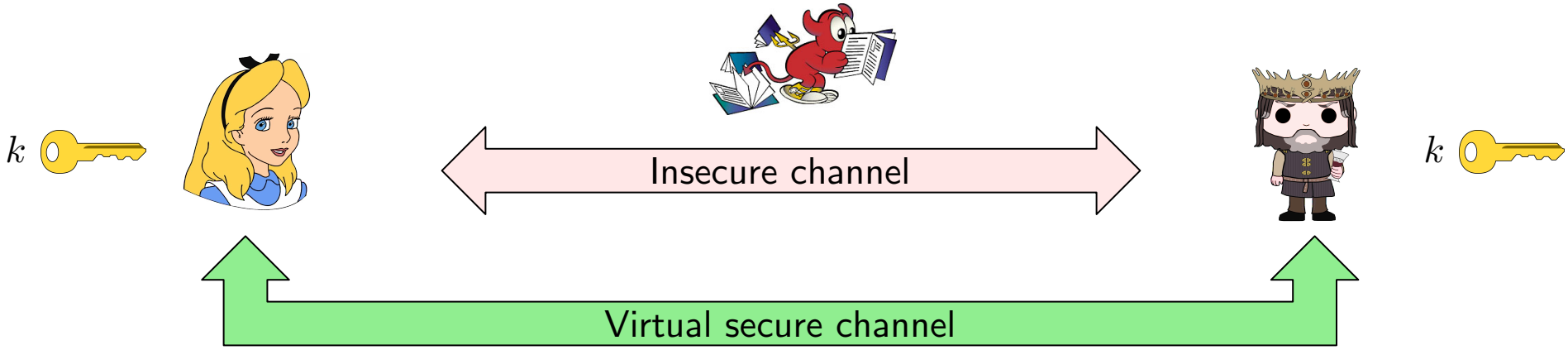



We have seen how Alice and Bob can use private-key criptography to establish a secure communication channel over an insecure one

To do this they need to share a secret key $k$

**How?**

# Key Distribution




We have seen how Alice and Bob can use private-key criptography to establish a secure communication channel over an insecure one

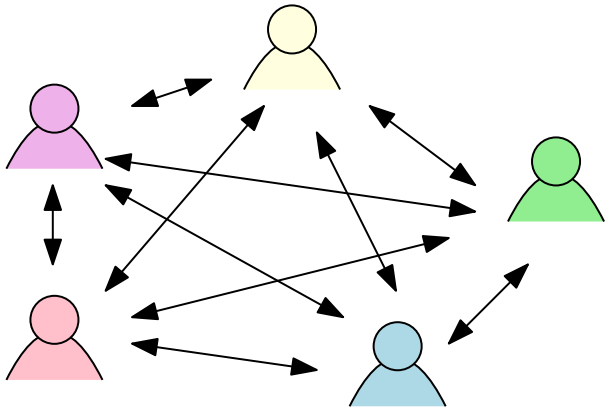
To do this they need to share a secret key $k$

**How?**

- Alice and Bob can meet in person
- Alice and Bob can use a **trusted** courier

This can be difficult (or even impossible) to do, depending on the situation

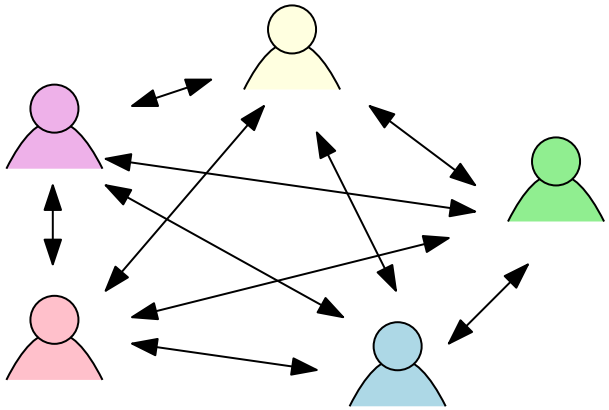
# Key Management



Imagine a big company with $N$ employees

- Any pair of employee might need to communicate (at some point)
- We want to ensure that the communication is confidential

# Key Management



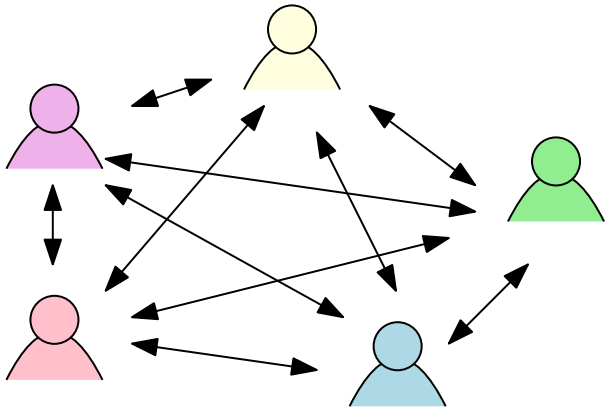
Imagine a big company with $N$ employees

- Any pair of employee might need to communicate (at some point)
- We want to ensure that the communication is confidential

A possible solution:

- Each pair of users $u, v$ generate their own secret key $k_{u,v}$

# Key Management



Imagine a big company with $N$ employees

- Any pair of employee might need to communicate (at some point)
- We want to ensure that the communication is confidential

A possible solution:

- Each pair of users $u, v$ generate their own secret key $k_{u,v}$
- There are $\Theta(N^2)$ keys in total
- If a new user joins the company, $\Theta(N)$ new keys are needed, and they need to be shared privately with each other user

**Not great...**

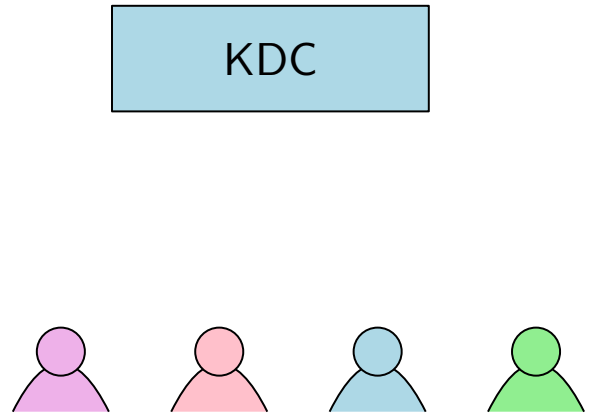
# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority
(e.g., the system administrator of the company)

# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority
(e.g., the system administrator of the company)

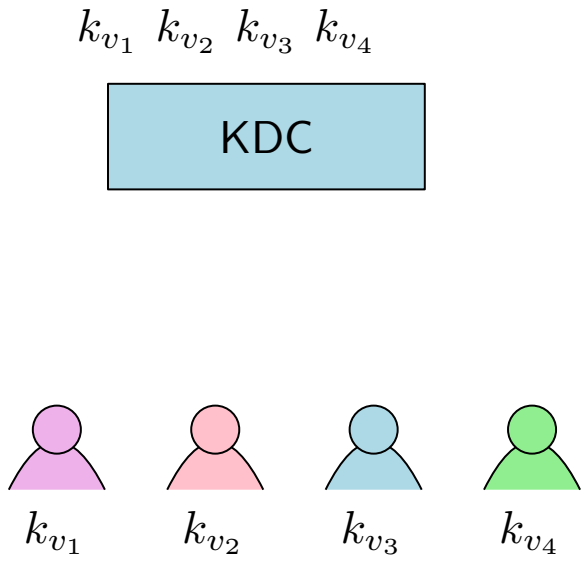
- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)



# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

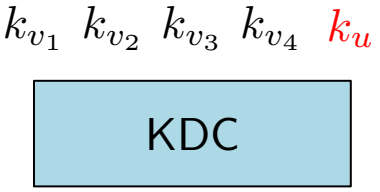
- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$

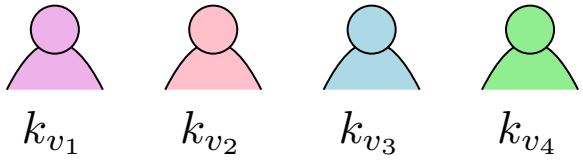


# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:

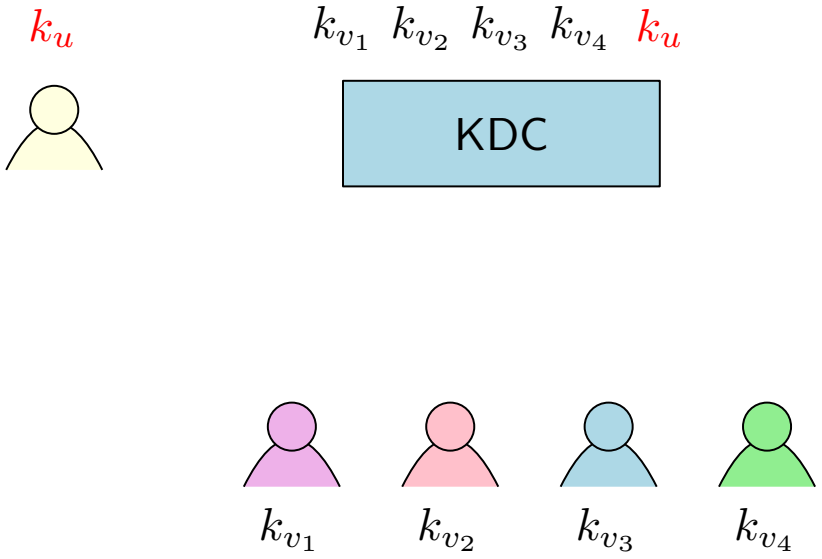



# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

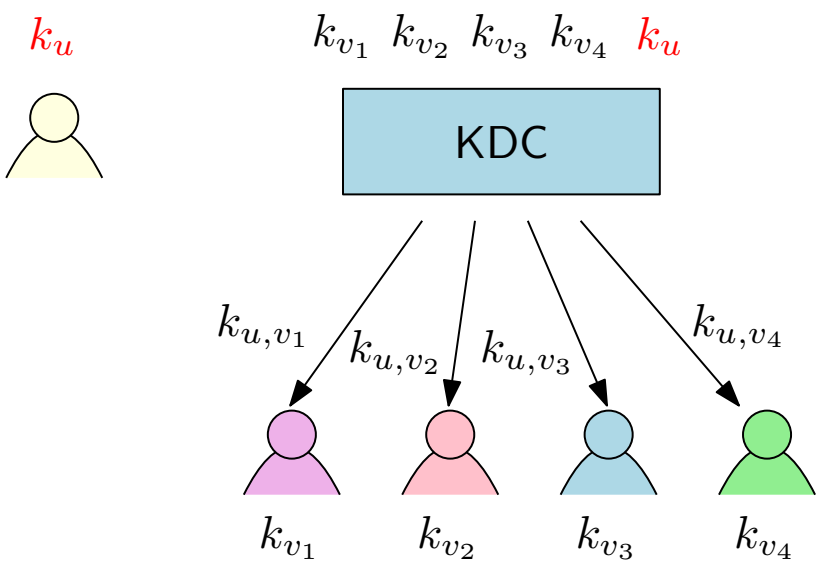
- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:
  - The KDC generates a key $k_{u,v}$ for each user $v \neq u$



# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

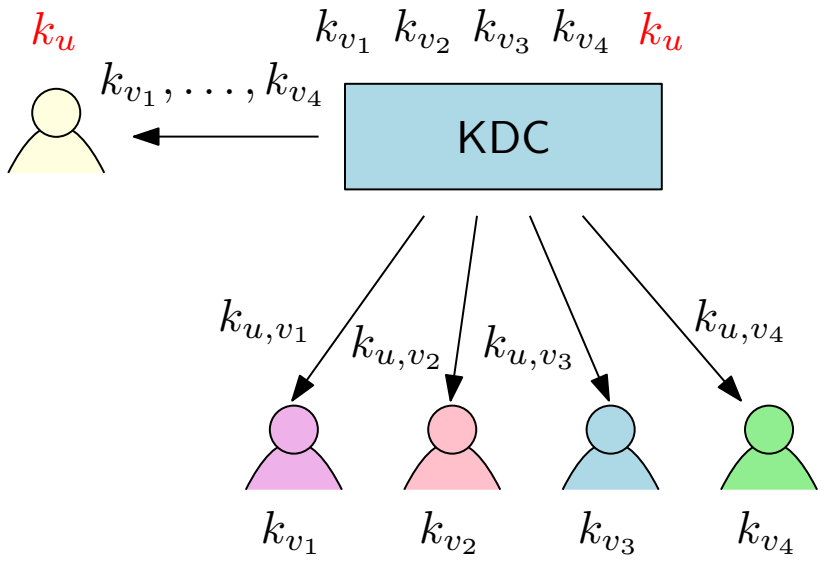
- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:
  - The KDC generates a key $k_{u,v}$ for each user $v \neq u$
  - The KDC encrypts each $k_{u,v}$ with $k_v$ and sends it to $v$



# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

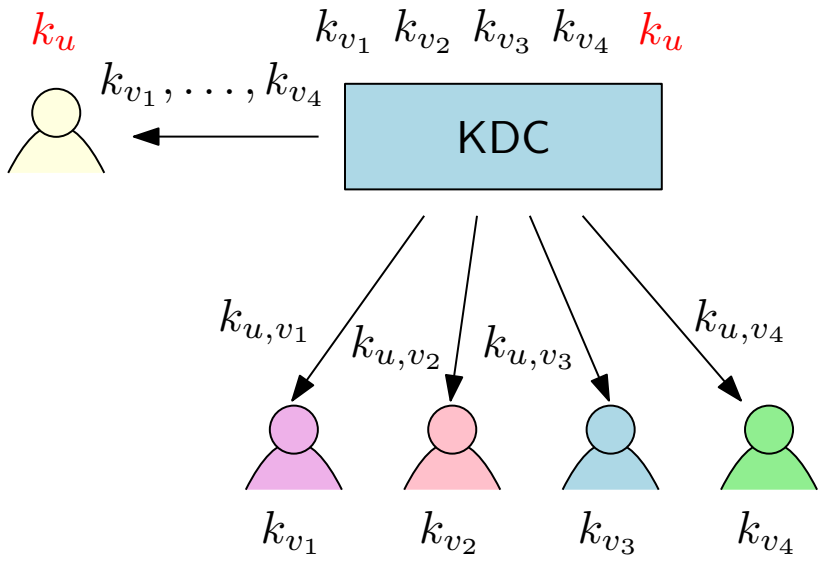
- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:
  - The KDC generates a key $k_{u,v}$ for each user $v \neq u$
  - The KDC encrypts each $k_{u,v}$ with $k_v$ and sends it to $v$
  - The KDC sends all keys $k_{u,v}$ to $u$ (encrypting them with $k_u$)



# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:
  - The KDC generates a key $k_{u,v}$ for each user $v \neq u$
  - The KDC encrypts each $k_{u,v}$ with $k_v$ and sends it to $v$
  - The KDC sends all keys $k_{u,v}$ to $u$ (encrypting them with $k_u$)

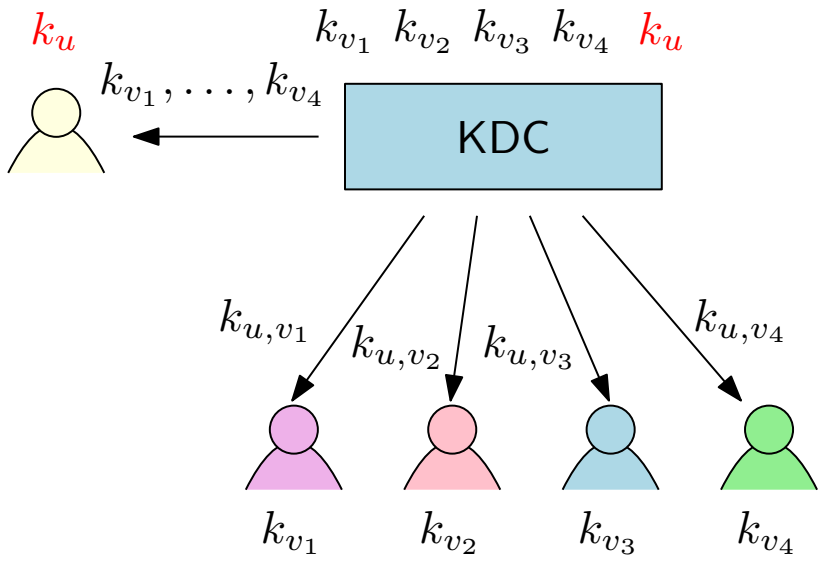



**Pro:** The new user $u$ does not need to privately meet all other $N$ employees to generate a shared key

# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:
  - The KDC generates a key $k_{u,v}$ for each user $v \neq u$
  - The KDC encrypts each $k_{u,v}$ with $k_v$ and sends it to $v$
  - The KDC sends all keys $k_{u,v}$ to $u$ (encrypting them with $k_u$)



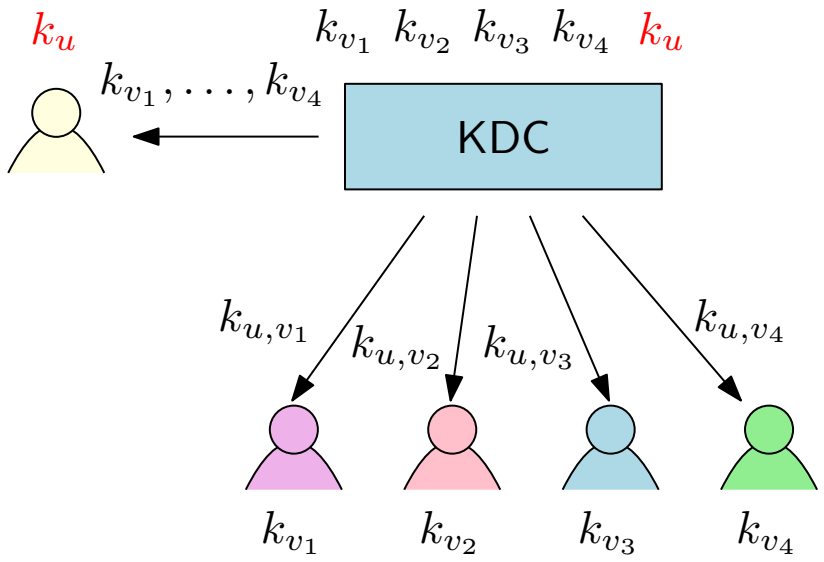

**Pro:** The new user $u$ does not need to privately meet all other $N$ employees to generate a shared key

**Con:** Everybody needs to trust the KDC

# A Partial Solution: Key-Distribution Centers ("offline solution")

Suppose that all the entities trust some central authority (e.g., the system administrator of the company)

- The (server set-up by the) system administrator can act as a **key-distribution center** (KDC)
- The KDC shares a key $k_v$ with each user $v$
- When a new user $u$ joins, they secretly exchange a key $k_u$ with the KDC. Then:
  - The KDC generates a key $k_{u,v}$ for each user $v \neq u$
  - The KDC encrypts each $k_{u,v}$ with $k_v$ and sends it to $v$
  - The KDC sends all keys $k_{u,v}$ to $u$ (encrypting them with $k_u$)




**Pro:** The new user $u$ does not need to privately meet all other $N$ employees to generate a shared key

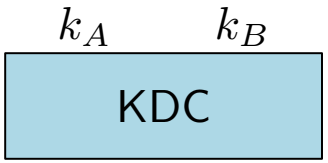
**Con:** Everybody needs to trust the KDC

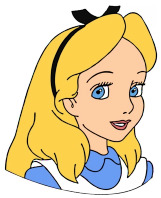
**Con:** Still $\Theta(N^2)$ keys overall

# A Partial Solution: Key-Distribution Centers ("online solution")

We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves



$k_A$

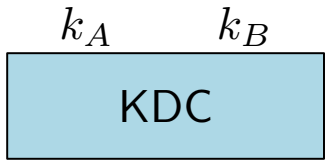
$k_B$

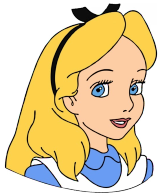
# A Partial Solution: Key-Distribution Centers ("online solution")

We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)



$k_A$

$k_B$

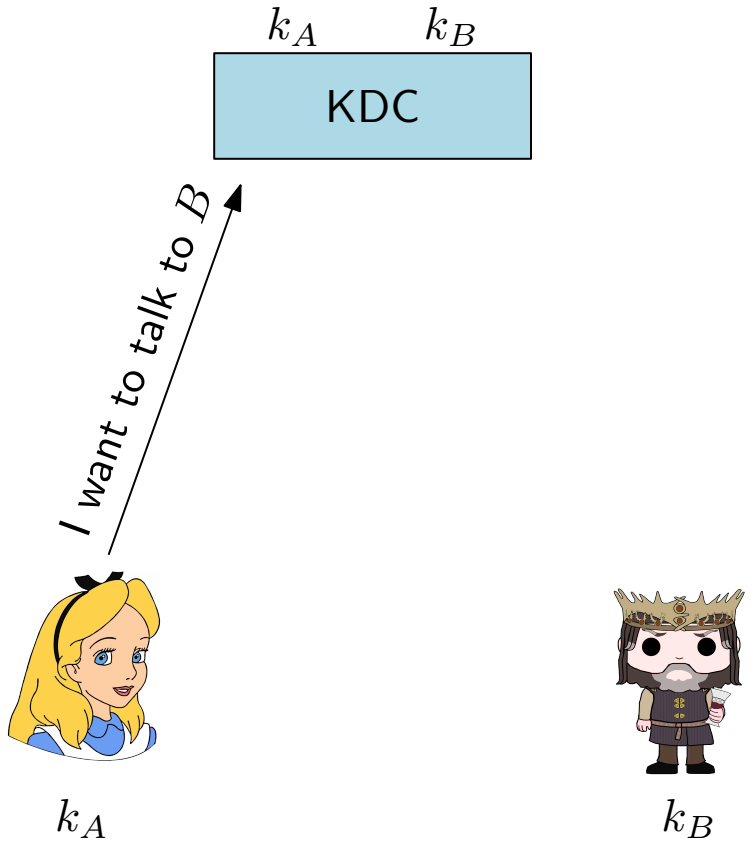
# A Partial Solution: Key-Distribution Centers ("online solution")

We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)



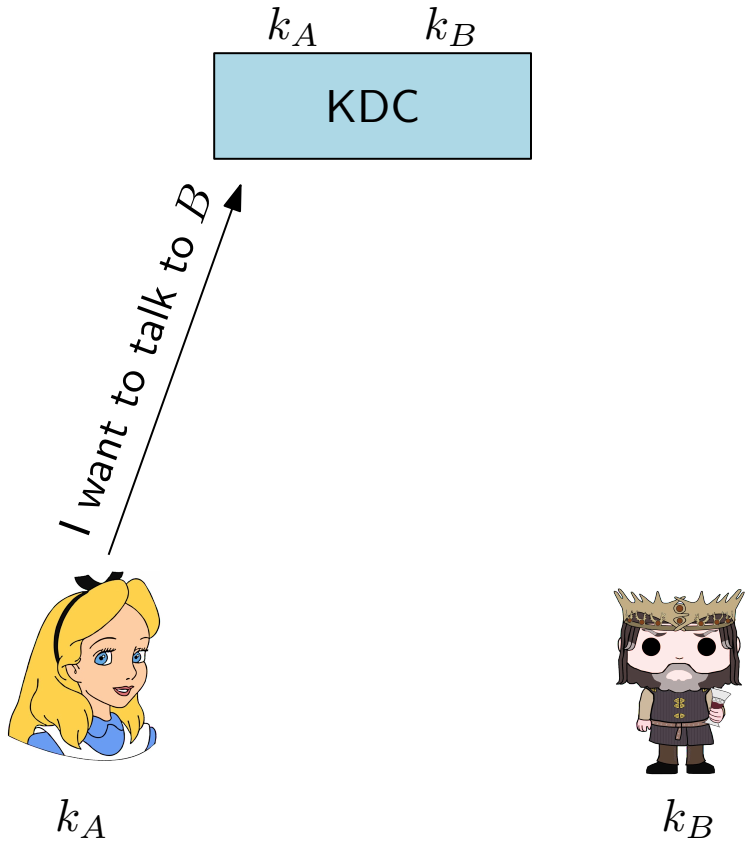
# A Partial Solution: Key-Distribution Centers ("online solution")

We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)
- The KDC generates a temporary *session key* $k$



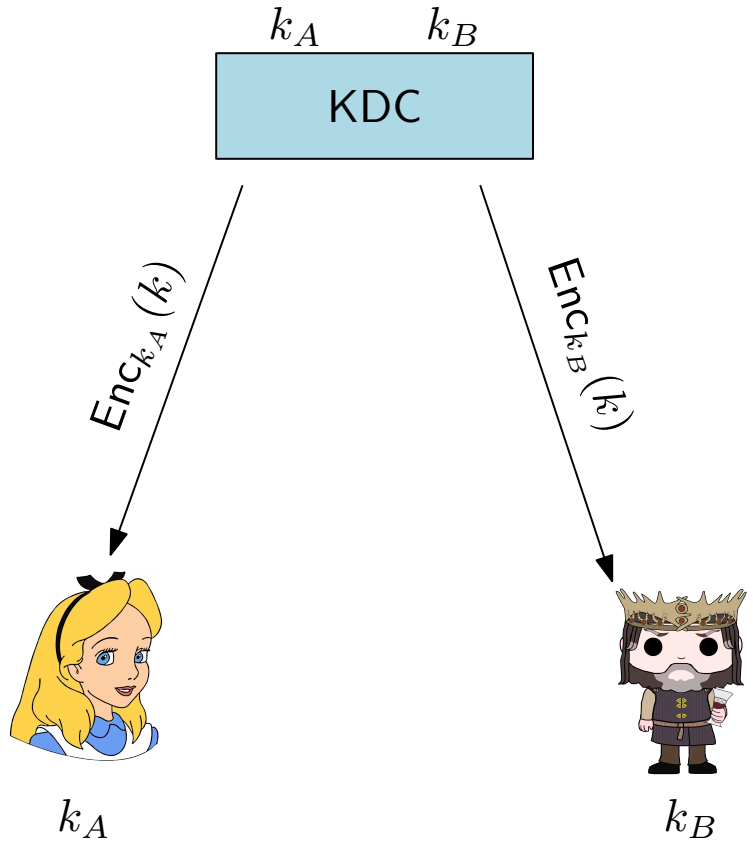
# A Partial Solution: Key-Distribution Centers ("online solution")

We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

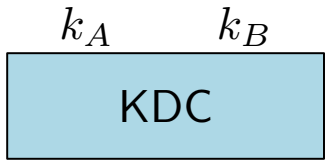
- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)
- The KDC generates a temporary *session key* $k$
- The KDC sends $k$ to $A$, encrypting the message with $k_A$
- The KDC sends $k$ to $B$, encrypting the message with $k_B$



# A Partial Solution: Key-Distribution Centers ("online solution")

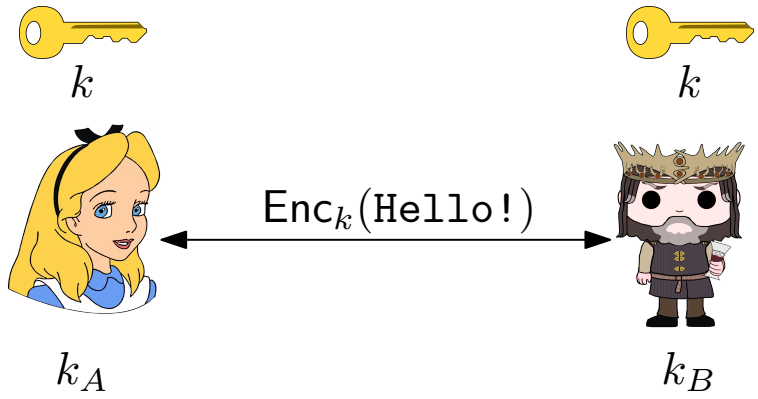
We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves




Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

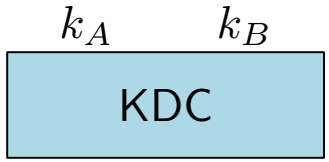
- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)
- The KDC generates a temporary *session key* $k$
- The KDC sends $k$ to $A$, encrypting the message with $k_A$
- The KDC sends $k$ to $B$, encrypting the message with $k_B$
- $A$ and $B$ start communicating using the session key $k$



# A Partial Solution: Key-Distribution Centers ("online solution")

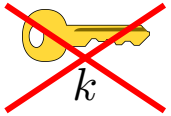
We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

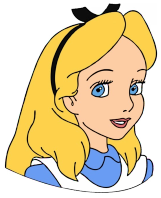
- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves



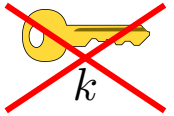

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)
- The KDC generates a temporary *session key* $k$
- The KDC sends $k$ to $A$, encrypting the message with $k_A$
- The KDC sends $k$ to $B$, encrypting the message with $k_B$
- $A$ and $B$ start communicating using the session key $k$
- At the end of the communication session, $A$ and $B$ destroy $k$



$k_A$

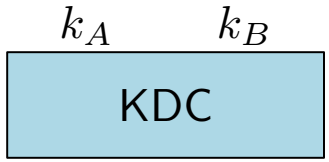



$k_B$

# A Partial Solution: Key-Distribution Centers ("online solution")

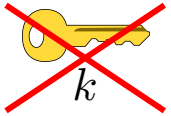
We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

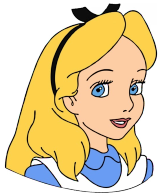
- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves



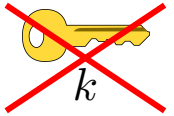

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)
- The KDC generates a temporary *session key* $k$
- The KDC sends $k$ to $A$, encrypting the message with $k_A$
- The KDC sends $k$ to $B$, encrypting the message with $k_B$
- $A$ and $B$ start communicating using the session key $k$
- At the end of the communication session, $A$ and $B$ destroy $k$



$k_A$



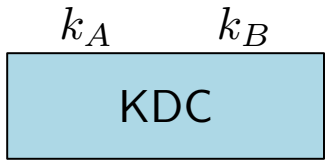

$k_B$

**Con:** Relies on the KDC being online (KDC can be replicated at the cost of more complexity)

# A Partial Solution: Key-Distribution Centers ("online solution")

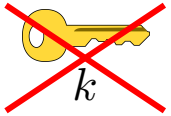
We can reduce the number of keys to $\Theta(N)$ using the KDC in a "online" fashion

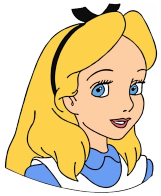
- Each user $v$ shares a key $k_v$ which the KDC
- Users do no share keys among themselves



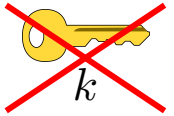

Suppose that a user $A$ (Alice) wants to talk to user $B$ (Bob)

- $A$ contacs the KDC and asks to talk to $B$
  (this message can be authenticated with a MAC, if desired)
- The KDC generates a temporary *session key* $k$
- The KDC sends $k$ to $A$, encrypting the message with $k_A$
- The KDC sends $k$ to $B$, encrypting the message with $k_B$
- $A$ and $B$ start communicating using the session key $k$
- At the end of the communication session, $A$ and $B$ destroy $k$



$k_A$




$k_B$

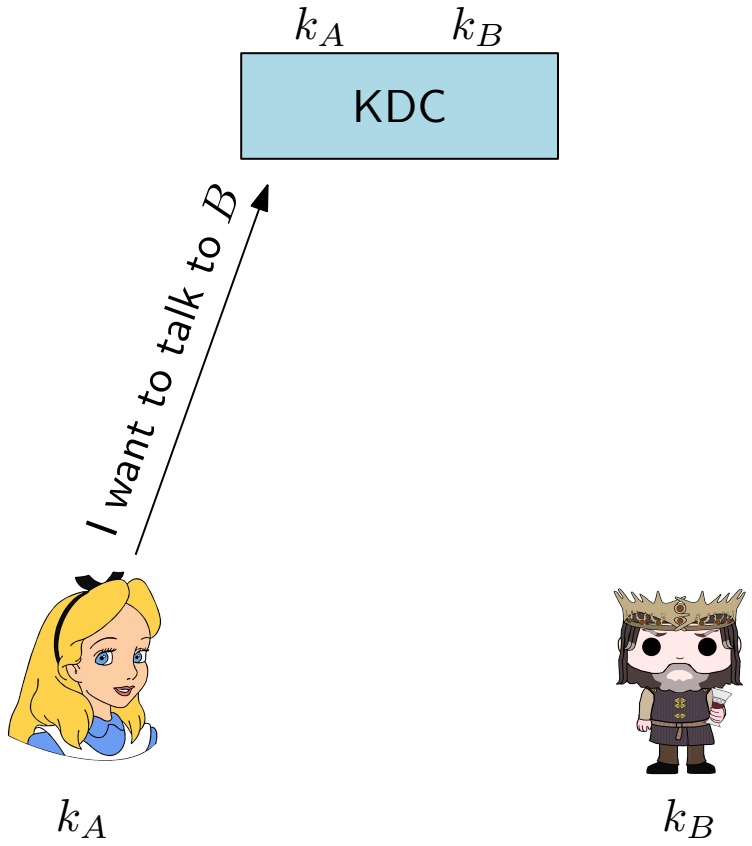
**Con:** Relies on the KDC being online (KDC can be replicated at the cost of more complexity)

**Con:** A data breach on the KDC compromises all communications

# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used

When $A$ contacts the KDC and asks to talk to $B$...

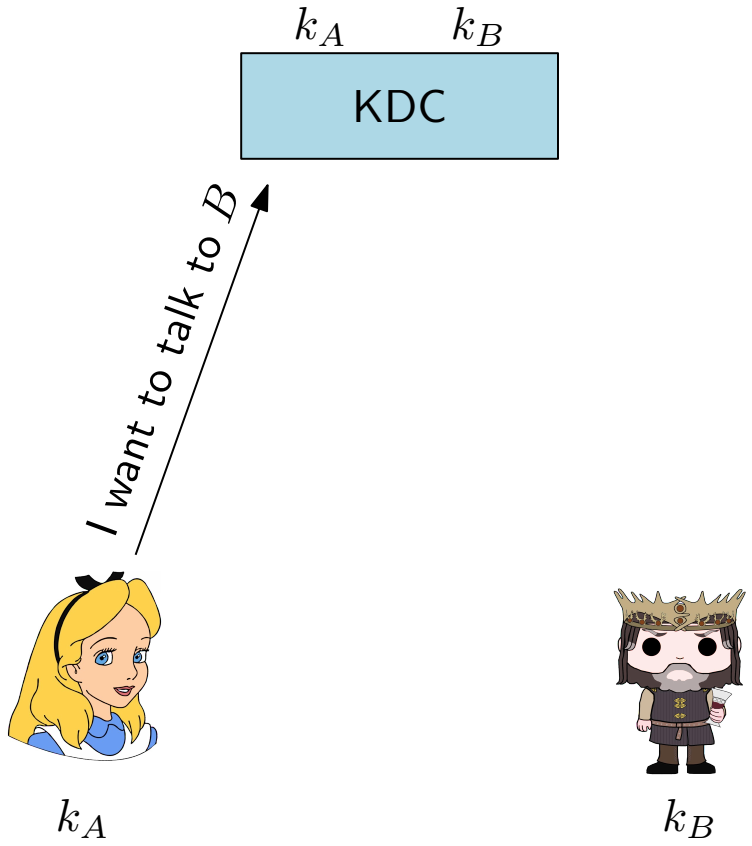


# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used

When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$

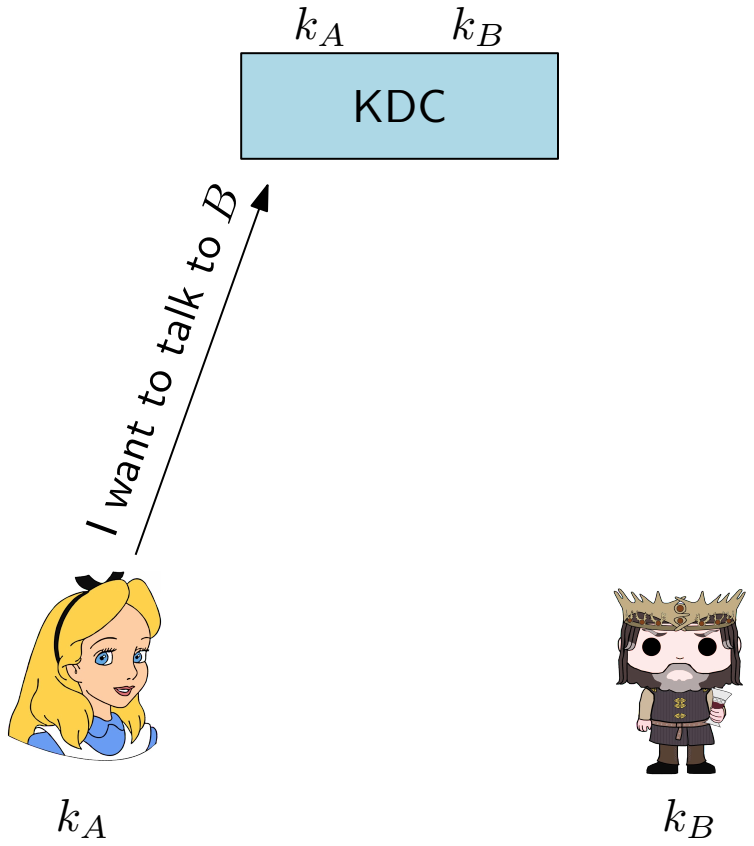


# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used

When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$

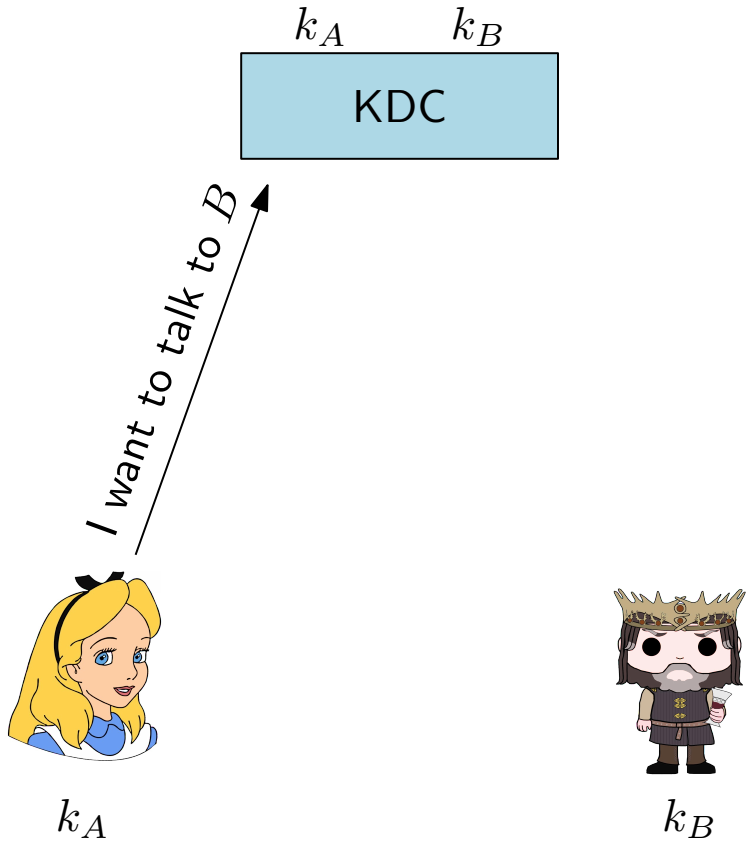


# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used

When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$
- The KDC encrypts $k$ with $k_B$ to obtain $k'_B \leftarrow \mathsf{Enc}_{k_B}(k)$. This is called a **ticket**.

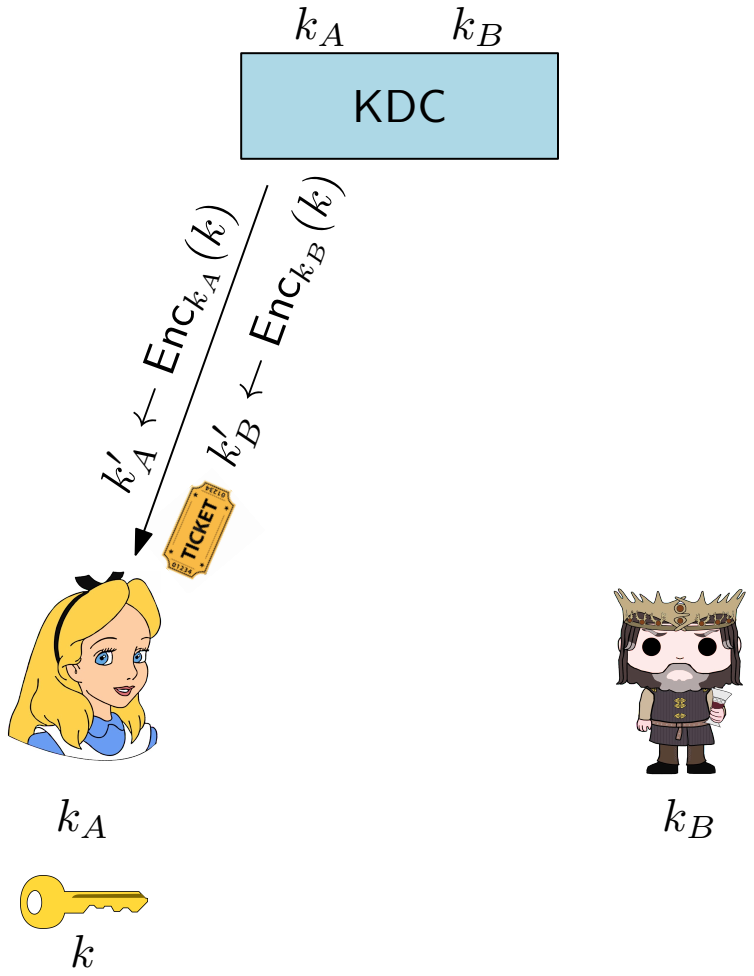


# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used

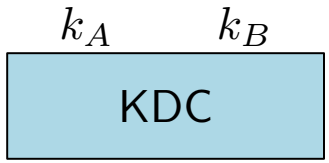
When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$
- The KDC encrypts $k$ with $k_B$ to obtain $k'_B \leftarrow \mathsf{Enc}_{k_B}(k)$. This is called a **ticket**.
- The KDC sends **both** $k'_A$ and the ticket to $A$



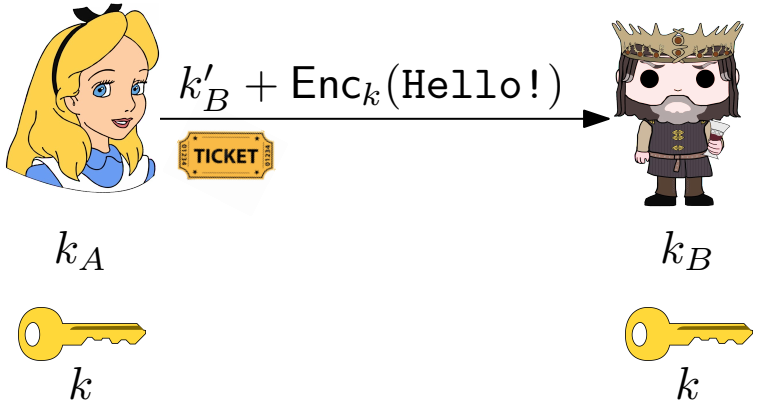
# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used



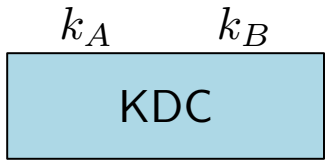

When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$
- The KDC encrypts $k$ with $k_B$ to obtain $k'_B \leftarrow \mathsf{Enc}_{k_B}(k)$. This is called a **ticket**.
- The KDC sends **both** $k'_A$ and the ticket to $A$
- $A$ initiates the communication by sending the ticket to $B$



# A Partial Solution: Key-Distribution Centers, an optimization

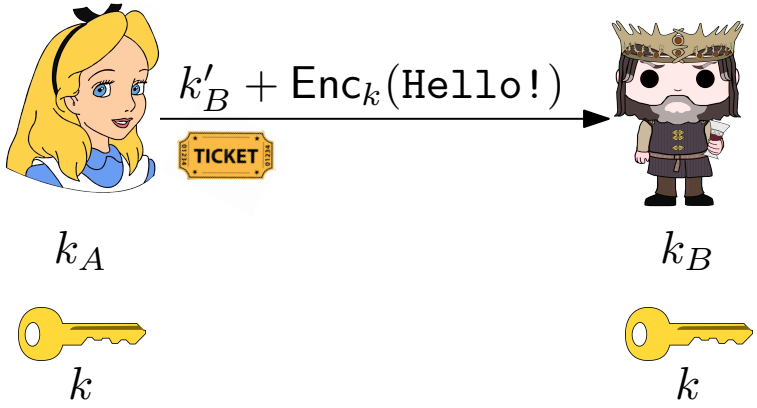
In practice, a slightly different approach is used




When $A$ contacts the KDC and asks to talk to $B$...

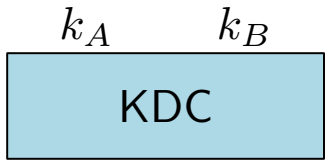
- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$
- The KDC encrypts $k$ with $k_B$ to obtain $k'_B \leftarrow \mathsf{Enc}_{k_B}(k)$. This is called a **ticket**.
- The KDC sends **both** $k'_A$ and the ticket to $A$
- $A$ initiates the communication by sending the ticket to $B$

Alternatively, the ticket can be retained by $A$ to initiate the communication with $B$ at a later point in time



# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used

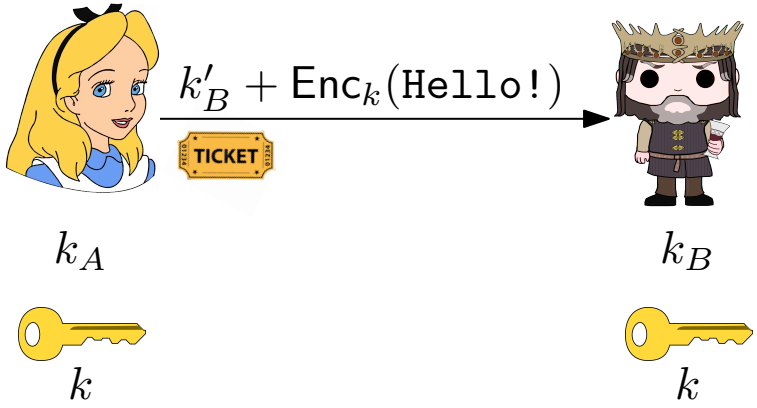



When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$
- The KDC encrypts $k$ with $k_B$ to obtain $k'_B \leftarrow \mathsf{Enc}_{k_B}(k)$. This is called a **ticket**.
- The KDC sends **both** $k'_A$ and the ticket to $A$
- $A$ initiates the communication by sending the ticket to $B$

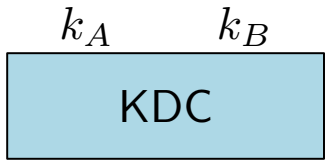
Alternatively, the ticket can be retained by $A$ to initiate the communication with $B$ at a later point in time

The ticket can contain a timestamp, after which it *expires*



# A Partial Solution: Key-Distribution Centers, an optimization

In practice, a slightly different approach is used



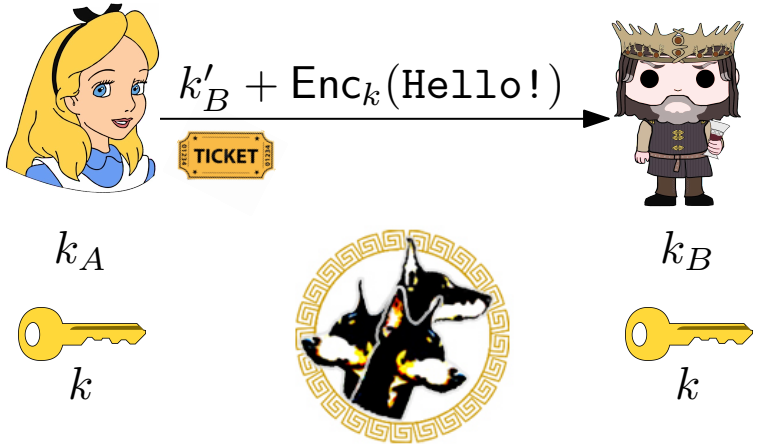

When $A$ contacts the KDC and asks to talk to $B$...

- The KDC generates a session key $k$
- The KDC encrypts $k$ with $k_A$ to obtain $k'_A \leftarrow \mathsf{Enc}_{k_A}(k)$
- The KDC encrypts $k$ with $k_B$ to obtain $k'_B \leftarrow \mathsf{Enc}_{k_B}(k)$. This is called a **ticket**.
- The KDC sends **both** $k'_A$ and the ticket to $A$
- $A$ initiates the communication by sending the ticket to $B$

Alternatively, the ticket can be retained by $A$ to initiate the communication with $B$ at a later point in time

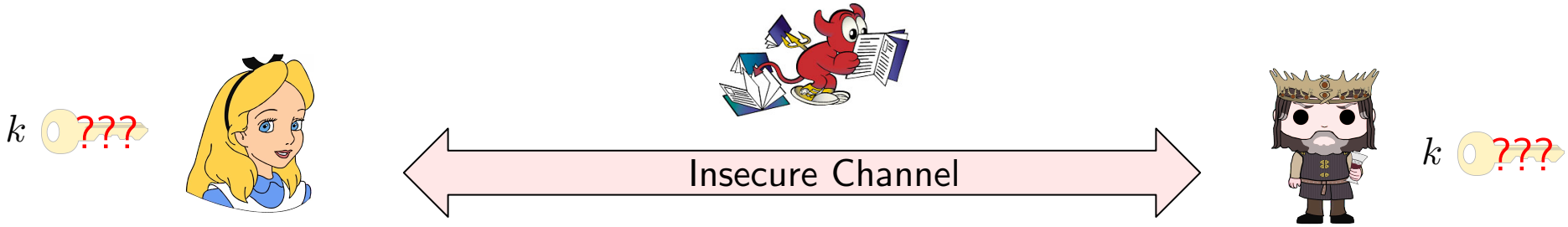
The ticket can contain a timestamp, after which it *expires*

This approach is used, e.g., in the Kerberos protocol.



# Open systems

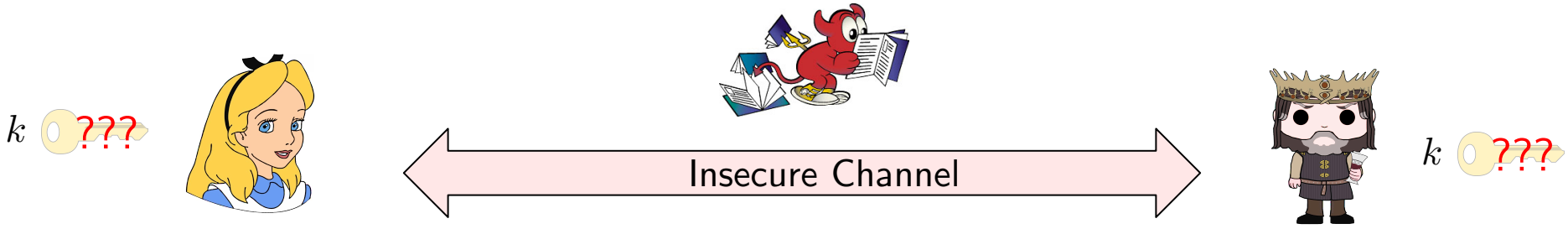
Alice e Bob want to communicate securely over an insecure channel




- But they did not know each other until now
- They don't have any common secret information
- No way to share a secret key in advance

# Open systems

Alice e Bob want to communicate securely over an insecure channel




- But they did not know each other until now
- They don't have any common secret information
- No way to share a secret key in advance

This happens all the time

- Sending an instant-message to someone for the first time
- Visiting a new website

**How do we handle this case?**

# Key Exchange

Classical private-key cryptography does not offer a satisfactory solution to these problems

# Key Exchange

Classical private-key cryptography does not offer a satisfactory solution to these problems

**Idea:**

- Some problem exhibit asymmetry
- Easy to compute, hard to invert (Candidate examples?)

# Key Exchange

Classical private-key cryptography does not offer a satisfactory solution to these problems

**Idea:**

- Some problem exhibit asymmetry
- Easy to compute, hard to invert (Candidate examples?)
- Use asymmetry in a clever way, to enable two parties to agree on a secret key over an insecure channel

# Key Exchange: Informal Security Definition

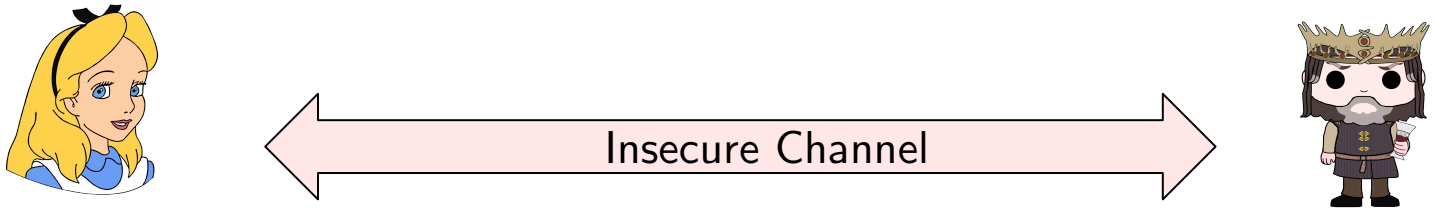
How do we formalize the concept of security for key exchanges?

# Key Exchange: Informal Security Definition

How do we formalize the concept of security for key exchanges?

**Idea:**

- Alice and Bob talk over the insecure channel following some (randomized) **key exchange protocol** (using a common shared input $1^n$ representing the security parameter in unary)

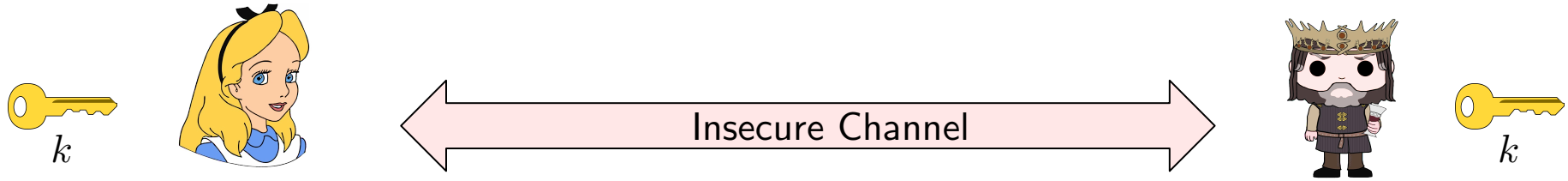


# Key Exchange: Informal Security Definition

How do we formalize the concept of security for key exchanges?

**Idea:**

- Alice and Bob talk over the insecure channel following some (randomized) **key exchange protocol** (using a common shared input $1^n$ representing the security parameter in unary)



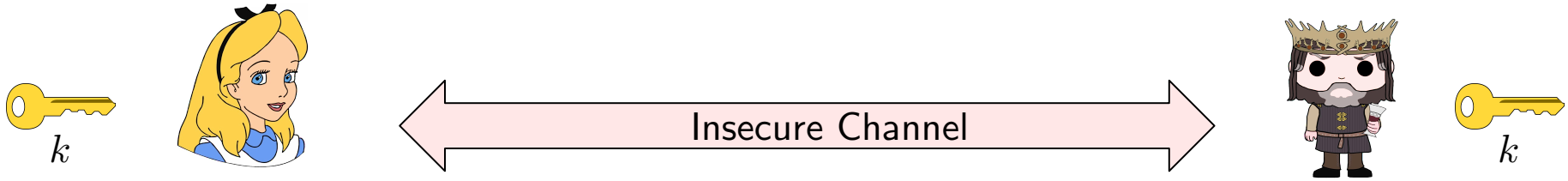

- At the end of the protocol, both Alice and Bob know some shared key $k$

# Key Exchange: Informal Security Definition

How do we formalize the concept of security for key exchanges?

**Idea:**

- Alice and Bob talk over the insecure channel following some (randomized) **key exchange protocol** (using a common shared input $1^n$ representing the security parameter in unary)



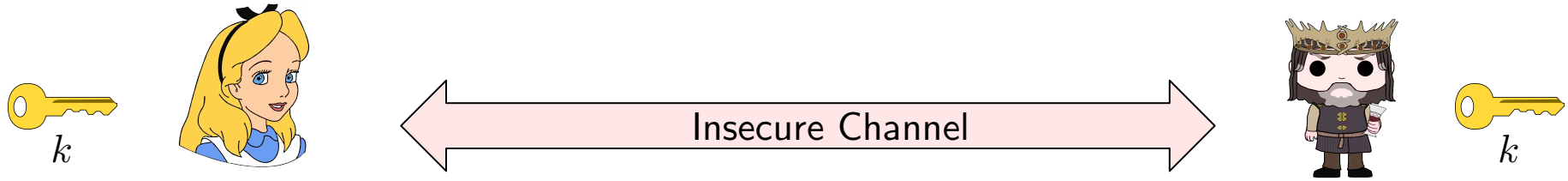

- At the end of the protocol, both Alice and Bob know some shared key $k$
- An adversary observes a full transcript of the conversation

# Key Exchange: Informal Security Definition

How do we formalize the concept of security for key exchanges?

**Idea:**

- Alice and Bob talk over the insecure channel following some (randomized) **key exchange protocol** (using a common shared input $1^n$ representing the security parameter in unary)




- At the end of the protocol, both Alice and Bob know some shared key $k$
- An adversary observes a full transcript of the conversation

**Security goal (informal):** even after observing the transcript, the shared key $k$ should be indistinguishable from a uniform key (to any polynomial-time adversary)

# The Key Exchange Experiment

Fix a key exchange protocol $\Pi$ and an attacker $\mathcal{A}$

We define the **key-exchange experiment** $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ as follows:

- The honest parties run $\Pi$ using $n$ as the security parameter.
- The interaction results in a transcript $\tau$ and in a shared key $k \in \{0,1\}^n$

# The Key Exchange Experiment

Fix a key exchange protocol $\Pi$ and an attacker $\mathcal{A}$

We define the **key-exchange experiment** $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ as follows:

- The honest parties run $\Pi$ using $n$ as the security parameter.
- The interaction results in a transcript $\tau$ and in a shared key $k \in \{0,1\}^n$

- A random bit $b$ is chosen u.a.r. from $\{0,1\}$.
  - If $b = 0$ then $k' \leftarrow k$.
  - Otherwise (when $b = 1$) $k'$ is chosen as a random uniform string from $\{0,1\}^n$.

# The Key Exchange Experiment

Fix a key exchange protocol $\Pi$ and an attacker $\mathcal{A}$

We define the **key-exchange experiment** $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ as follows:

- The honest parties run $\Pi$ using $n$ as the security parameter.
- The interaction results in a transcript $\tau$ and in a shared key $k \in \{0,1\}^n$

- A random bit $b$ is chosen u.a.r. from $\{0,1\}$.
  - If $b = 0$ then $k' \leftarrow k$.
  - Otherwise (when $b = 1$) $k'$ is chosen as a random uniform string from $\{0,1\}^n$.
- $\mathcal{A}$ is given $k'$ and the transcript $\tau$

- $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$

# The Key Exchange Experiment

Fix a key exchange protocol $\Pi$ and an attacker $\mathcal{A}$

We define the **key-exchange experiment** $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ as follows:

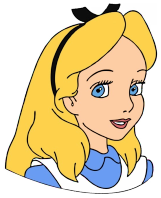
- The honest parties run $\Pi$ using $n$ as the security parameter.
- The interaction results in a transcript $\tau$ and in a shared key $k \in \{0,1\}^n$

- A random bit $b$ is chosen u.a.r. from $\{0,1\}$.
  - If $b = 0$ then $k' \leftarrow k$.
  - Otherwise (when $b = 1$) $k'$ is chosen as a random uniform string from $\{0,1\}^n$.
- $\mathcal{A}$ is given $k'$ and the transcript $\tau$

- $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$

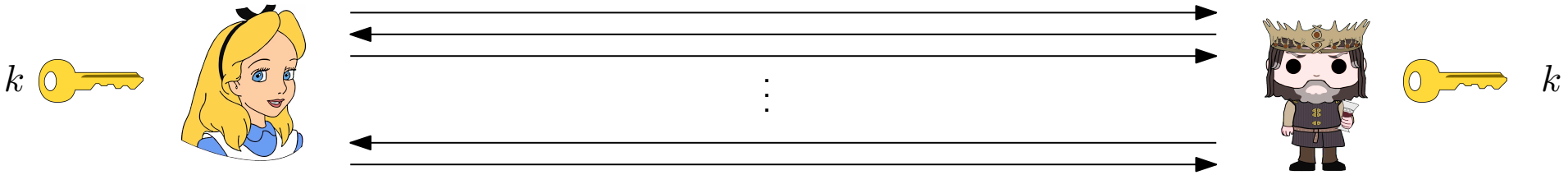
The outcome of the experiment is defined to be $1$ if $b' = b$ and $0$ otherwise

# The Key Exchange Experiment



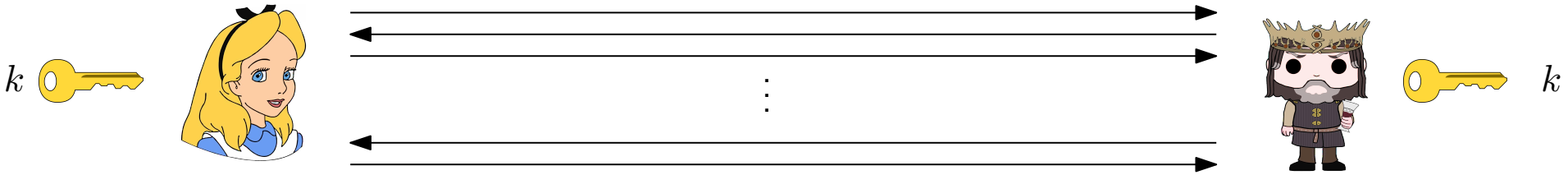
# The Key Exchange Experiment

Messages exchanged
following $\Pi$




# The Key Exchange Experiment

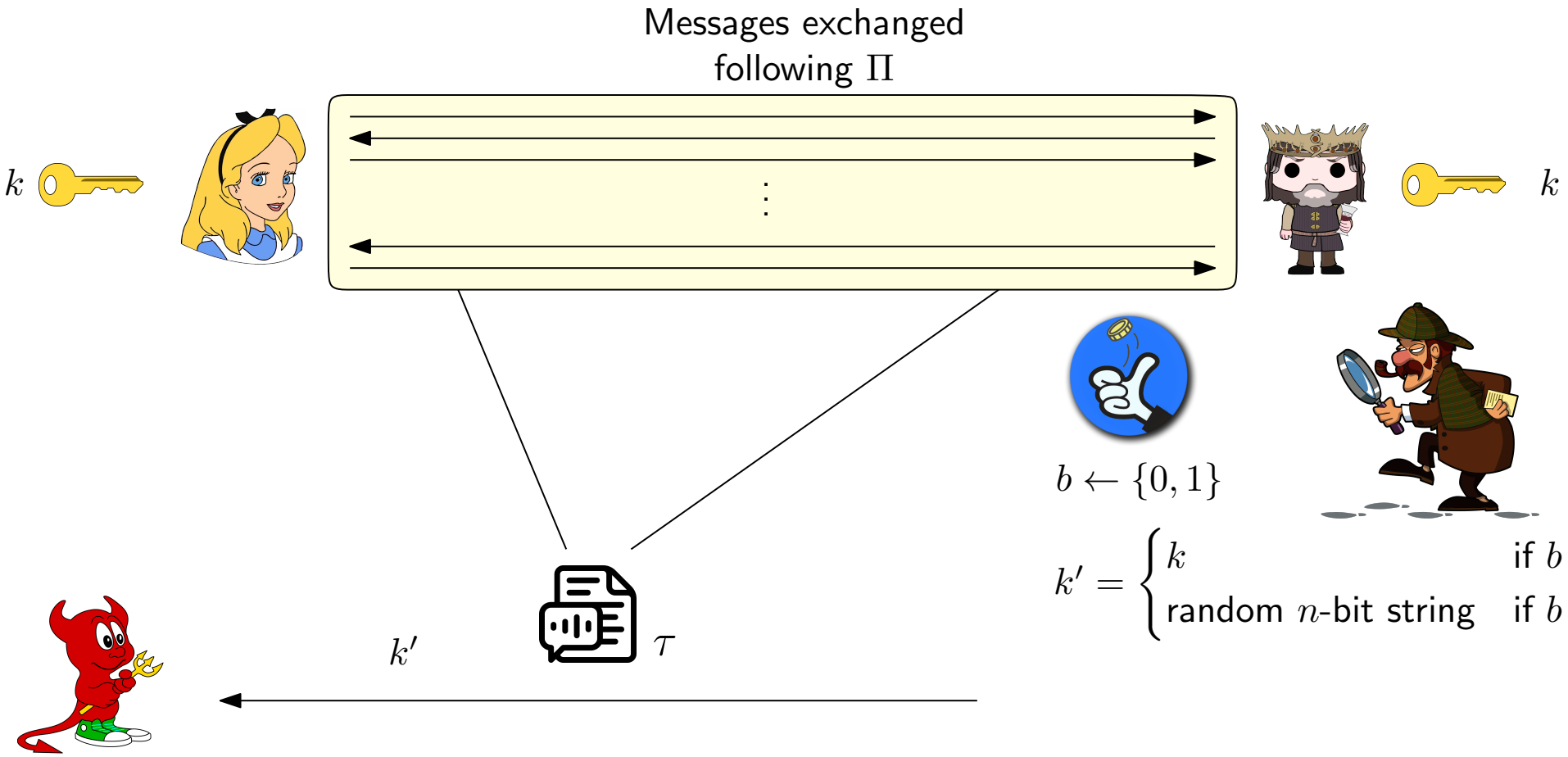
Messages exchanged
following $\Pi$



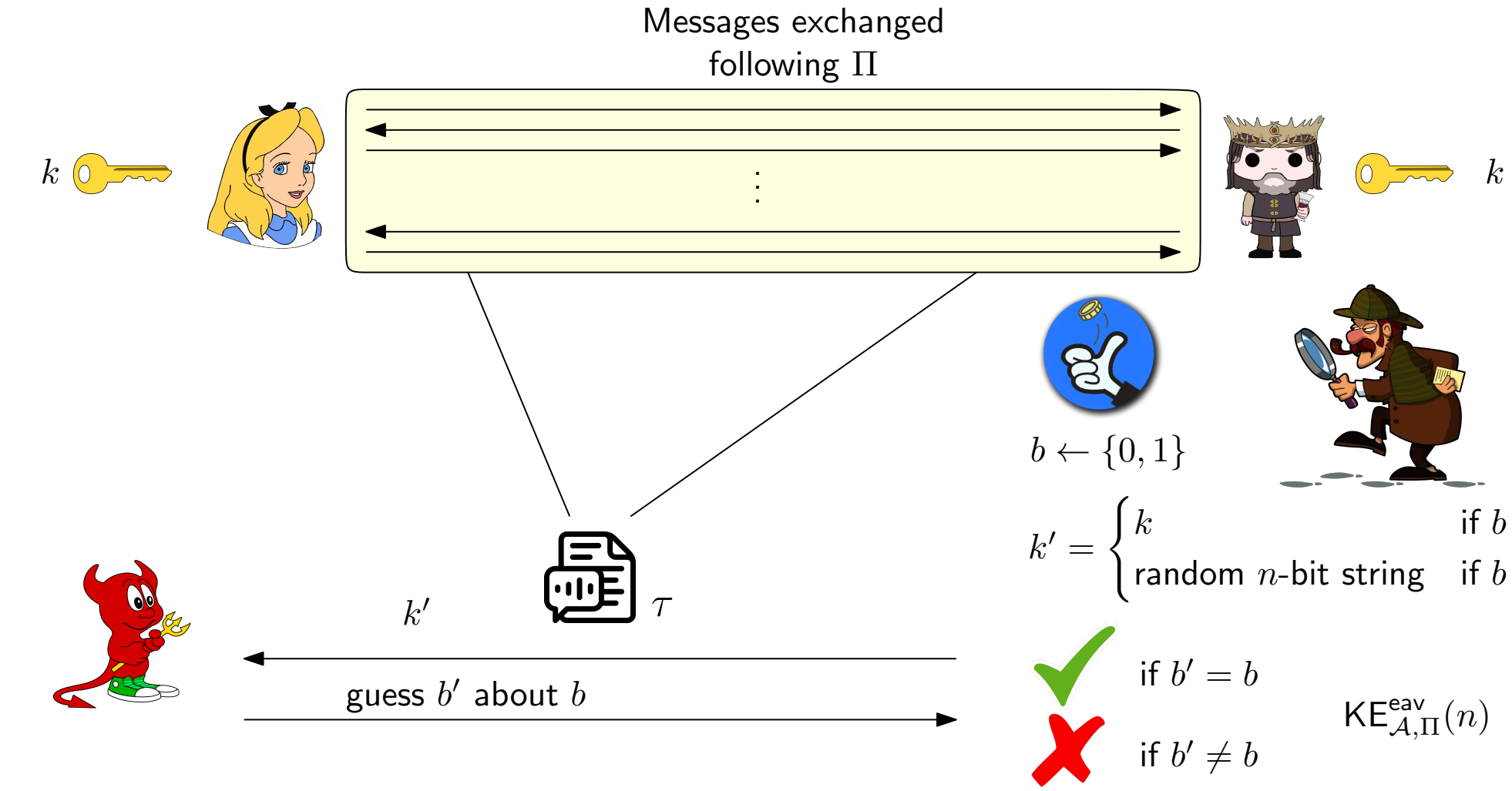

$b \leftarrow \{0,1\}$

$$k' = \begin{cases} k & \text{if } b = 0 \\ \text{random } n\text{-bit string} & \text{if } b = 1 \end{cases}$$

# The Key Exchange Experiment



# The Key Exchange Experiment



# Key Exchange: Formal Security Definition

**Definition:** A key-exchange protocol $\Pi$ is **secure in the presence of an eavesdropper** if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function $\varepsilon$ such that

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n).$$

# Key Exchange: Formal Security Definition

**Definition:** A key-exchange protocol $\Pi$ is **secure in the presence of an eavesdropper** if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function $\varepsilon$ such that

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n).$$

Notice that being unable to compute $k$ from the transcript $\tau$ is not a strong enough security guarantee

- The requirement we impose is **stronger**. Namely $k$ must look just like a random string.
- This is necessary since we are going to use $k$ for private-key cryptography.