AUTOMATIC VERIFICATION OF CRYPTOGRAPHIC PROTOCOLS

Privacy-type properties

Vincent Cheval
ENS Cachan, LSV

December 03, 2012

Context

Most communications take place over a public network

It is important to ensure their security

Cryptographic protocols

Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy)
- use cryptographic primitives (e.g. encryption, signature)

Cryptographic protocols

Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy)
- use cryptographic primitives (e.g. encryption, signature)

What do we need to build a secure protocol ?

Cryptographic protocols

Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy)
- use cryptographic primitives (e.g. encryption, signature)

What do we need to build a secure protocol ?

- Reliable cryptography
- Correct protocol specification
- Implementation satisfying the specification

Cryptographic protocols

Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy)
- use cryptographic primitives (e.g. encryption, signature)

What do we need to build a secure protocol ?

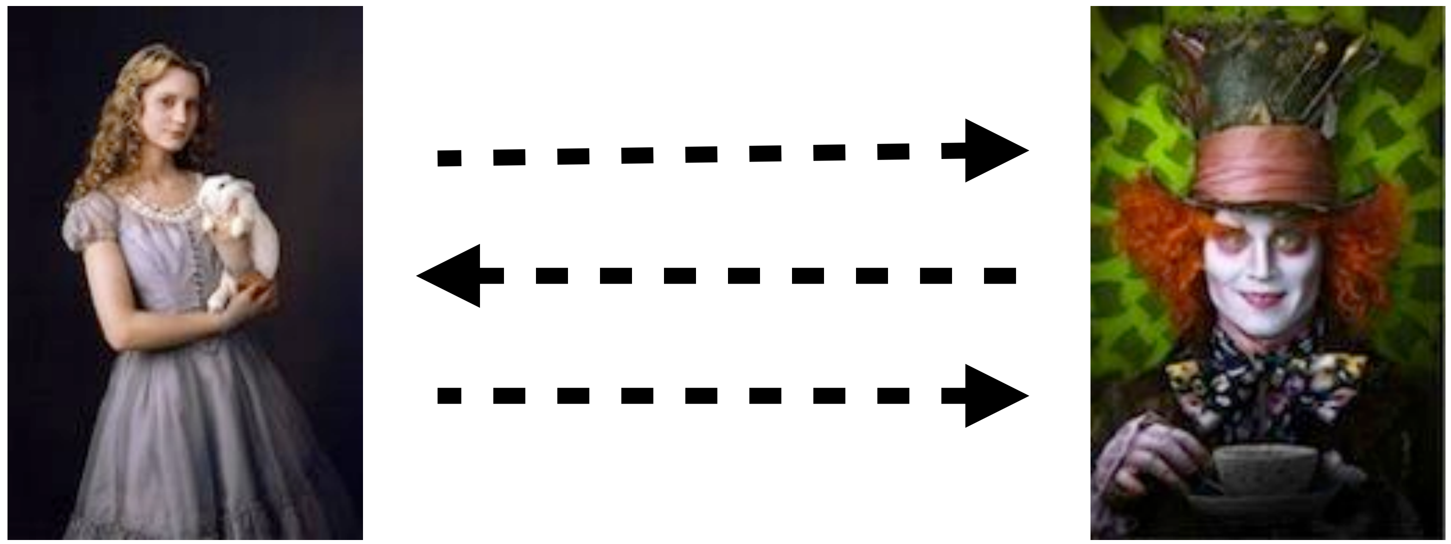
- Reliable cryptography
- Correct protocol specification
- Implementation satisfying the specification

Symbolic model

Alice

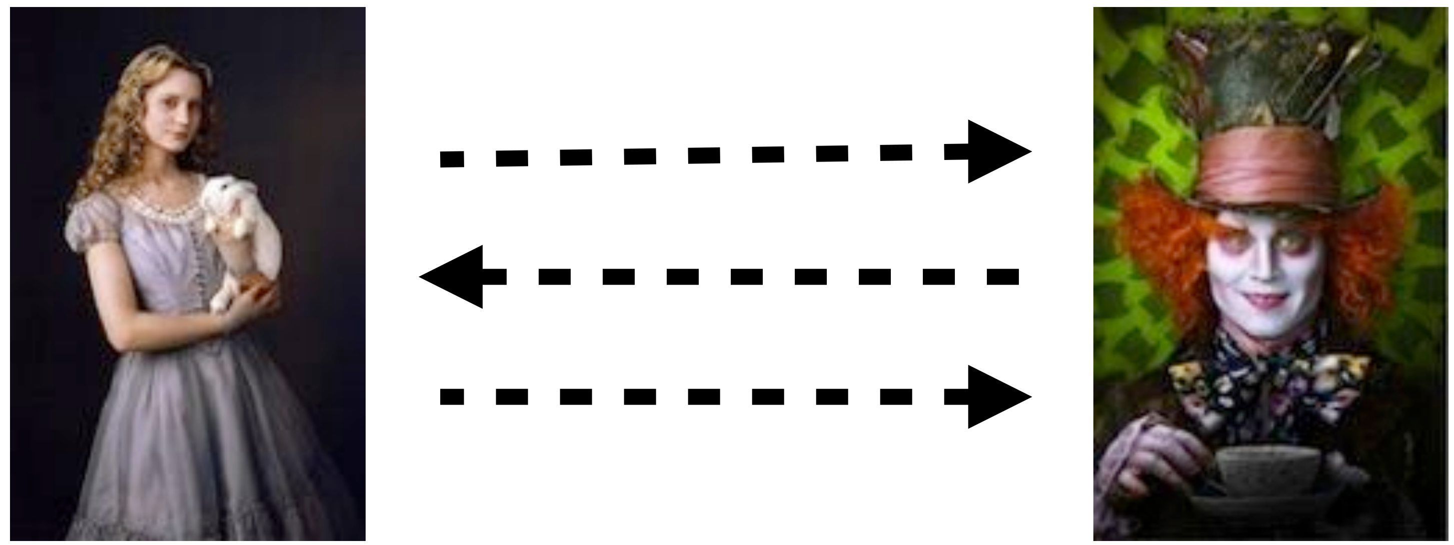
Bob

Symbolic model



Alice Bob

Symbolic model



Alice Bob

- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M\rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

Symbolic model

Alice

Attacker

Bob

- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M\rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

Symbolic model

Alice

Attacker

Bob

- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M\rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

The attacker can

- intercept all messages
- transmit or modify messages
- test equality between messages

Symbolic model

Alice

Attacker

Bob

- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M\rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

The attacker can

- intercept all messages
- transmit or modify messages
- test equality between messages

Symbolic model

Alice

Attacker

Bob

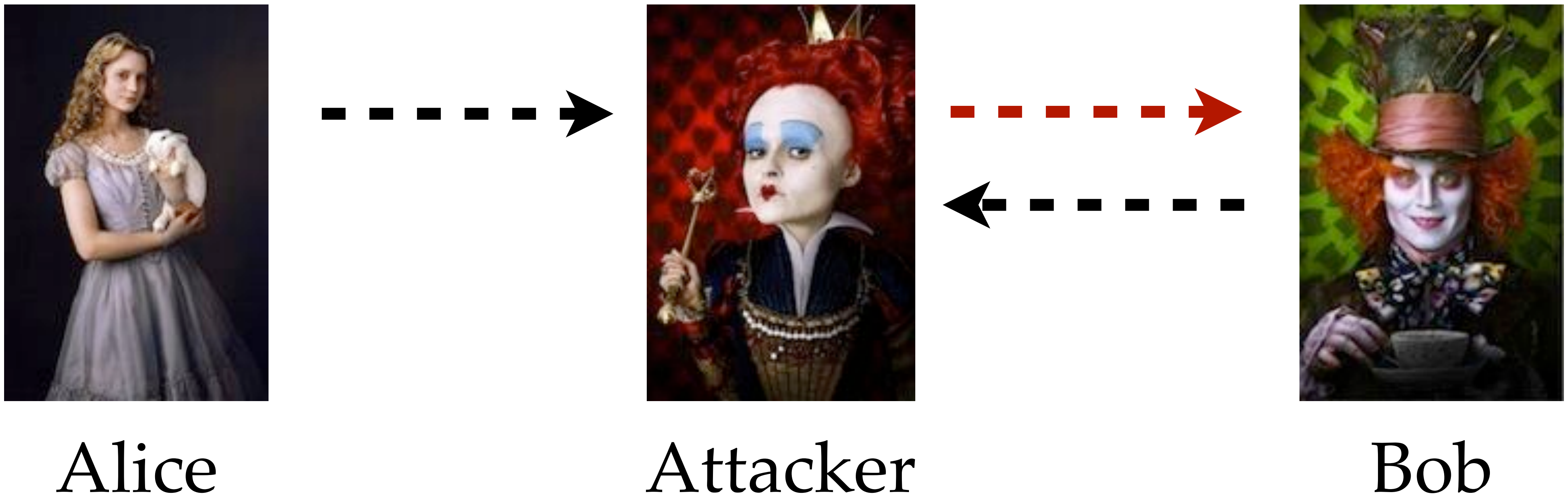
- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M \rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

The attacker can

- intercept all messages
- transmit or modify messages
- test equality between messages

Symbolic model



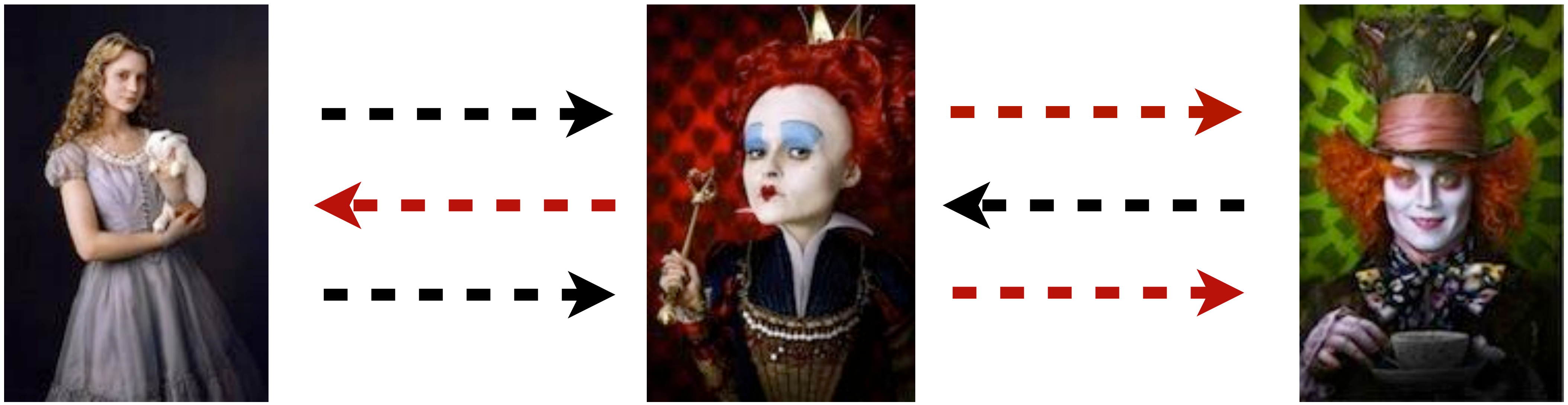
- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M \rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

The attacker can

- intercept all messages
- transmit or modify messages
- test equality between messages

Symbolic model



Alice Attacker Bob

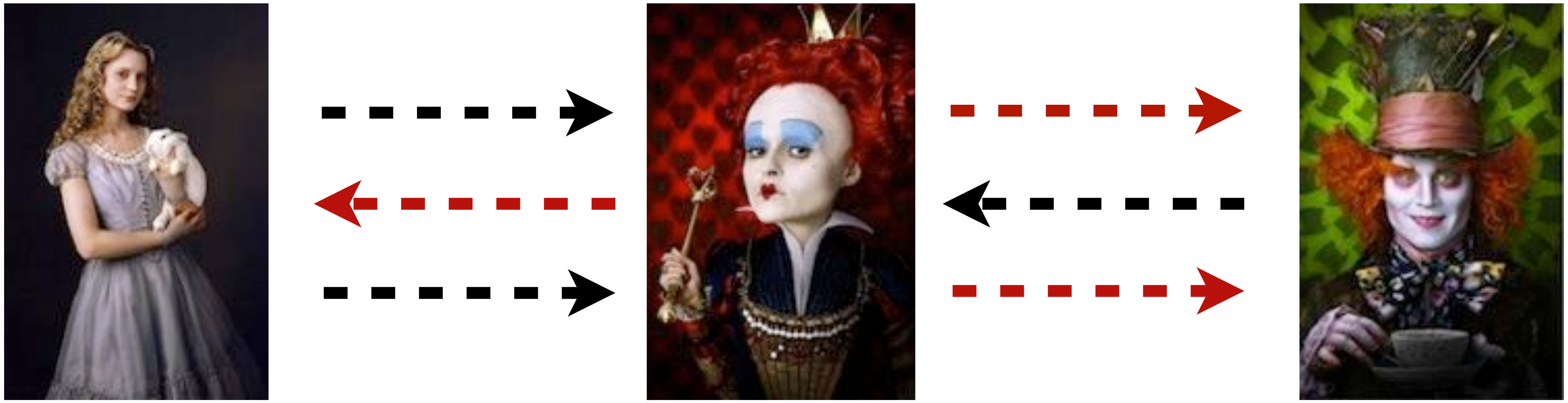
- We assume perfect cryptographic primitives
- Messages are represented by terms

$$\{N\}_{\mathsf{pk}(k)} \qquad \langle N, M \rangle \qquad \mathsf{adec}(\{N\}_{\mathsf{pk}(k)}, k)$$

The attacker can

- intercept all messages
- transmit or modify messages
- test equality between messages

Symbolic model



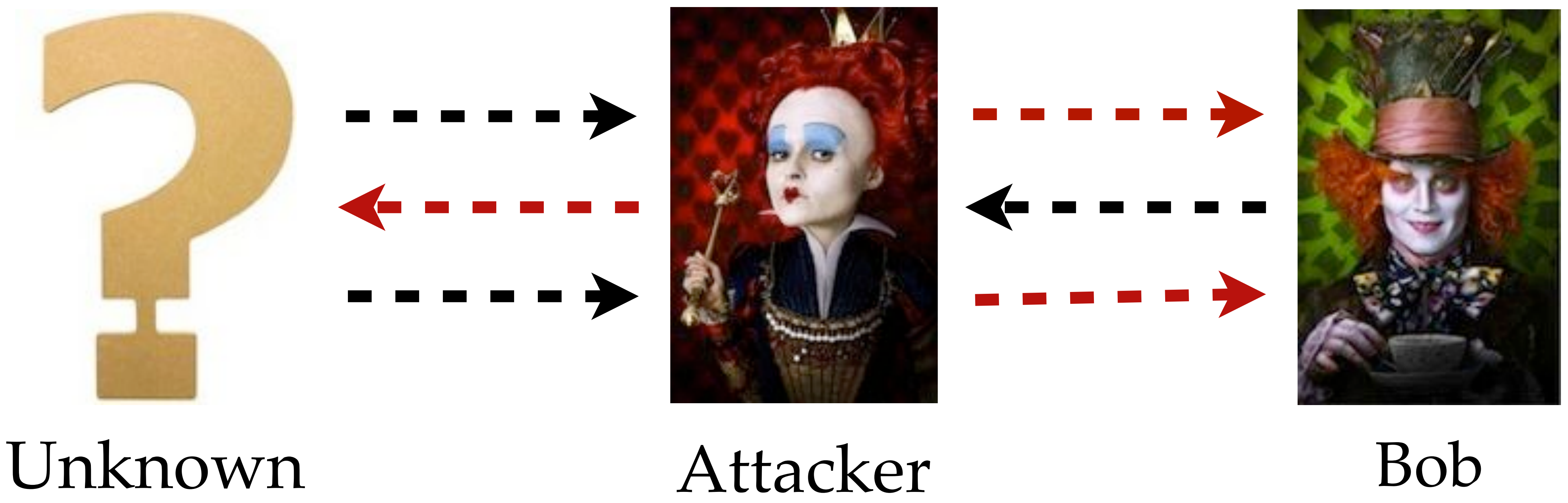
Alice Attacker Bob

Security properties:

- Reachability properties
- Equivalence properties

Security properties

Equivalence properties: anonymity

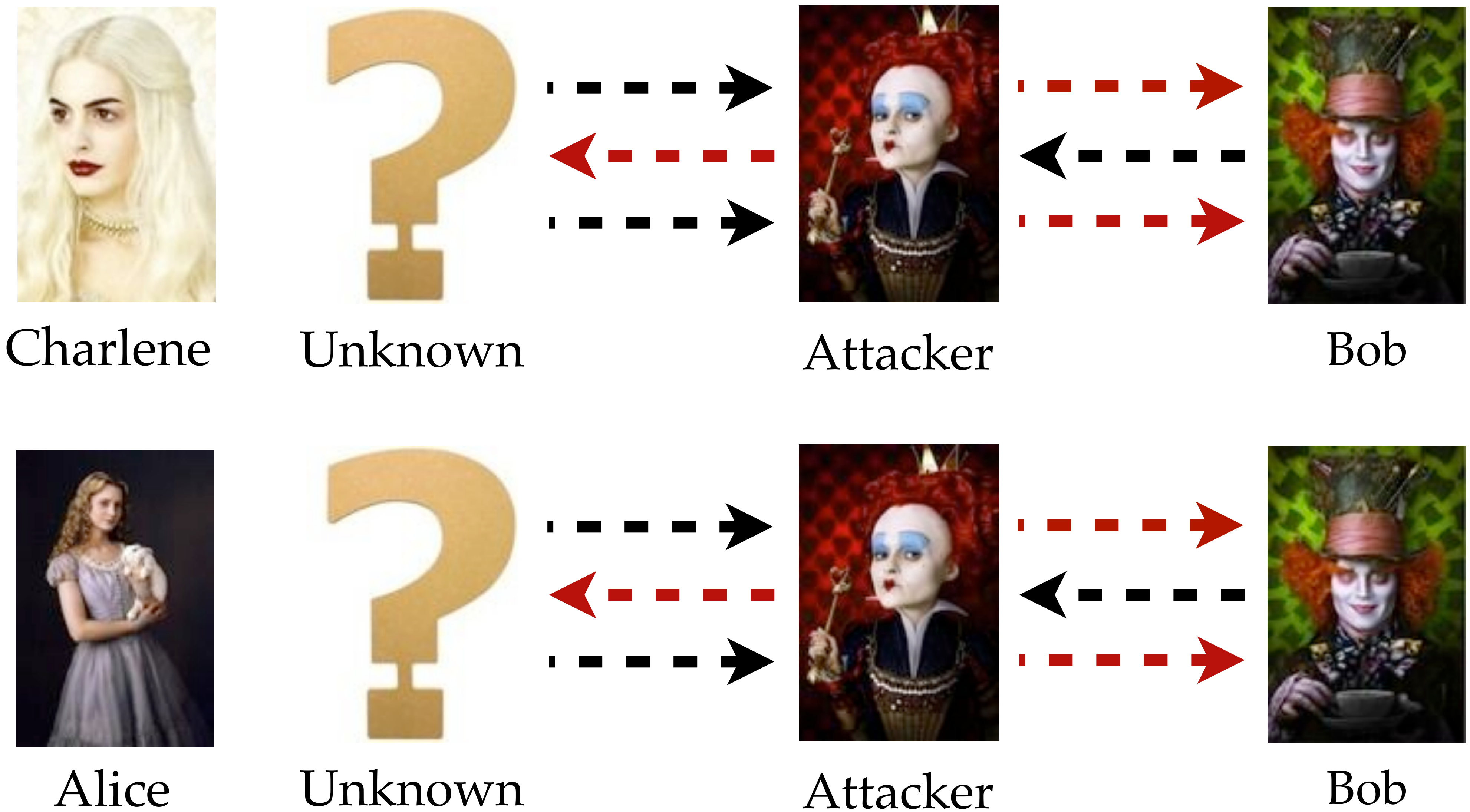


Security properties

Equivalence properties: anonymity

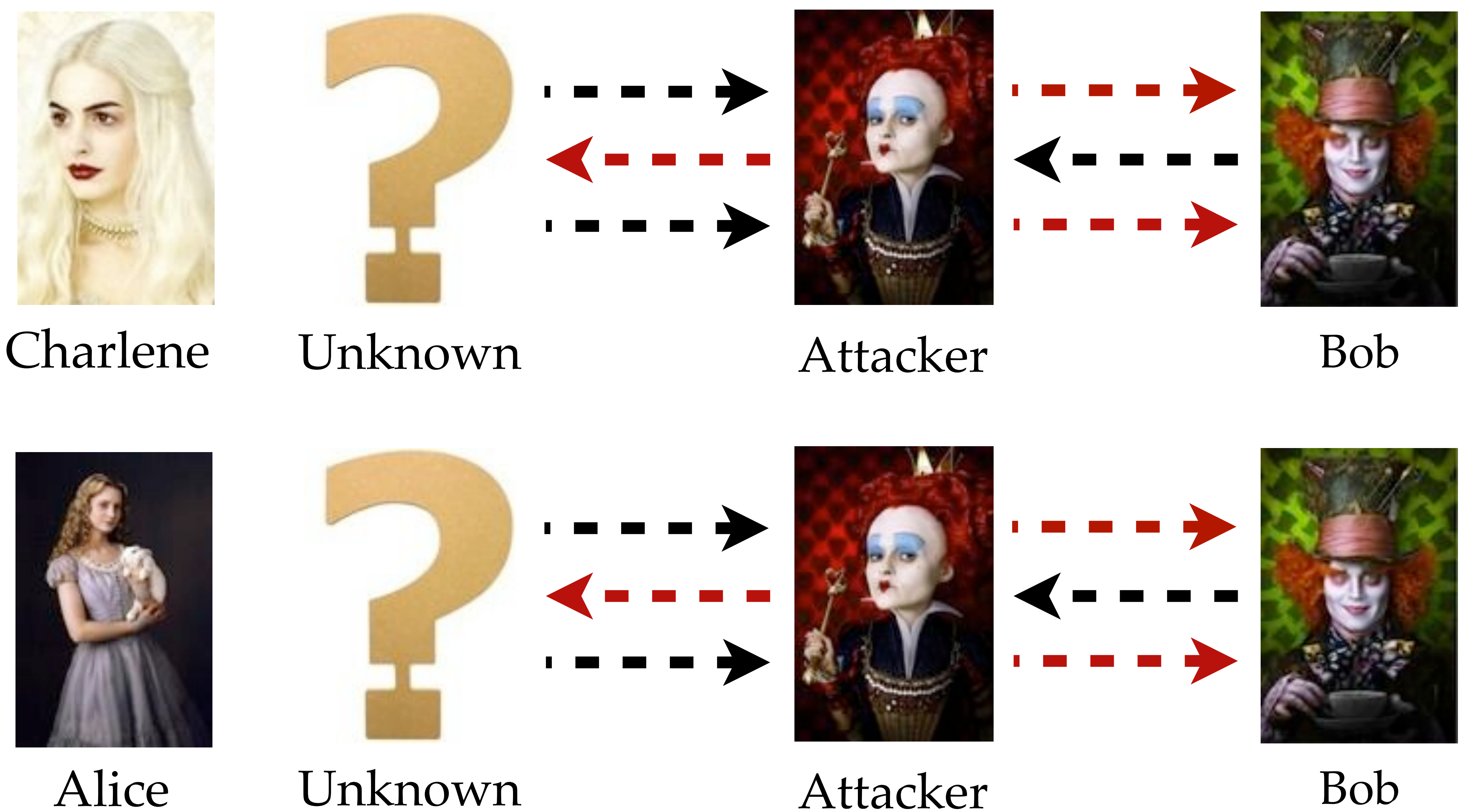
Security properties

Equivalence properties: anonymity



Security properties

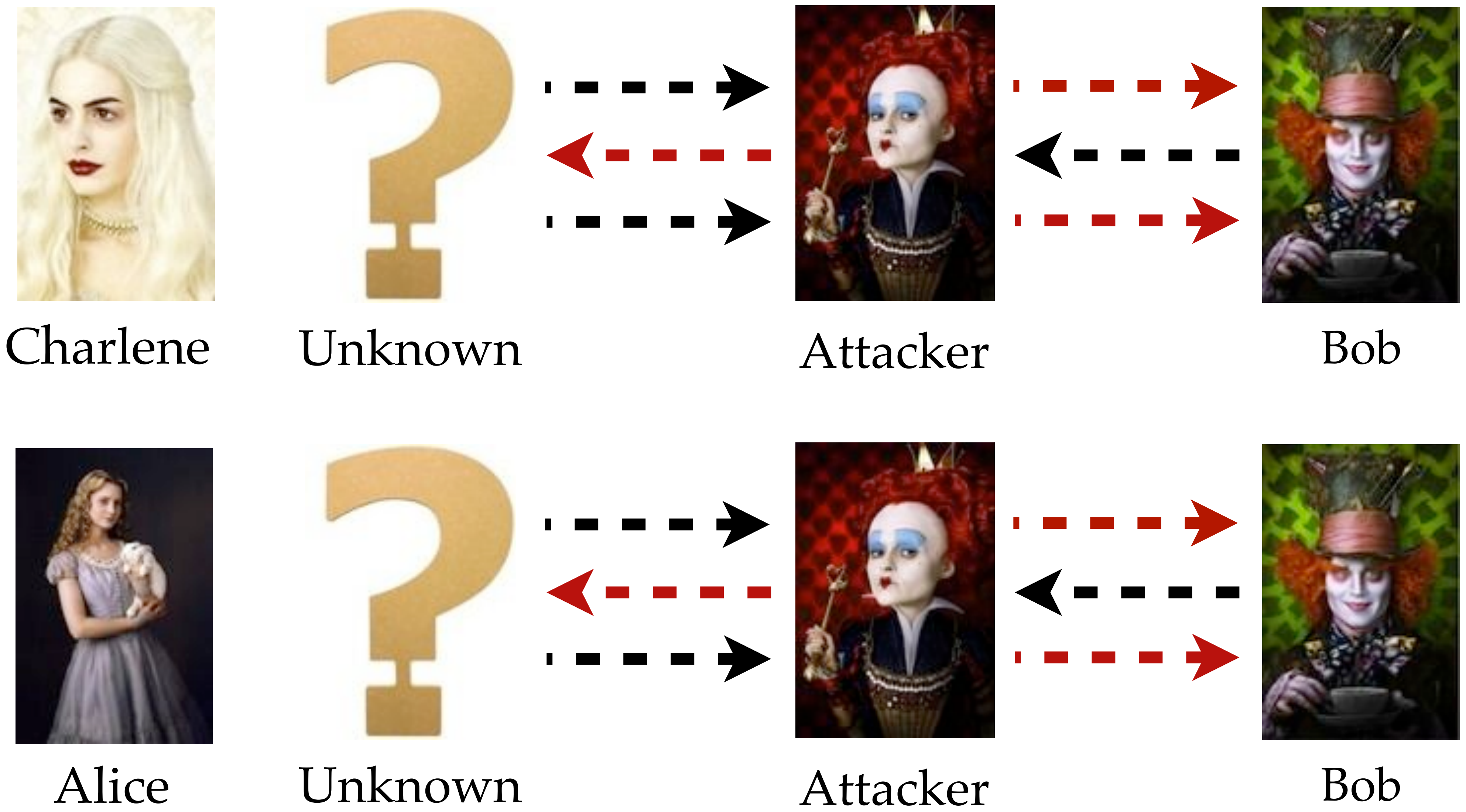
Equivalence properties: anonymity



Can the intruder distinguish the two situations ?

Security properties

Equivalence properties: anonymity



Trace equivalence

Examples

Private authentication protocol

Examples

Private authentication protocol

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

Alice

Bob

Examples

Private authentication protocol

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

Alice

$\mathsf{pk}(k_A)$?

Bob

Examples

Private authentication protocol

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle N_a, N_b, \mathsf{pk}(k_B)\rangle\}_{\mathsf{pk}(k_A)}$

$\mathsf{pk}(k_A)?$

Alice

Bob

Examples

Private authentication protocol

Alice

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle N_a, N_b, \mathsf{pk}(k_B)\rangle\}_{\mathsf{pk}(k_A)}$

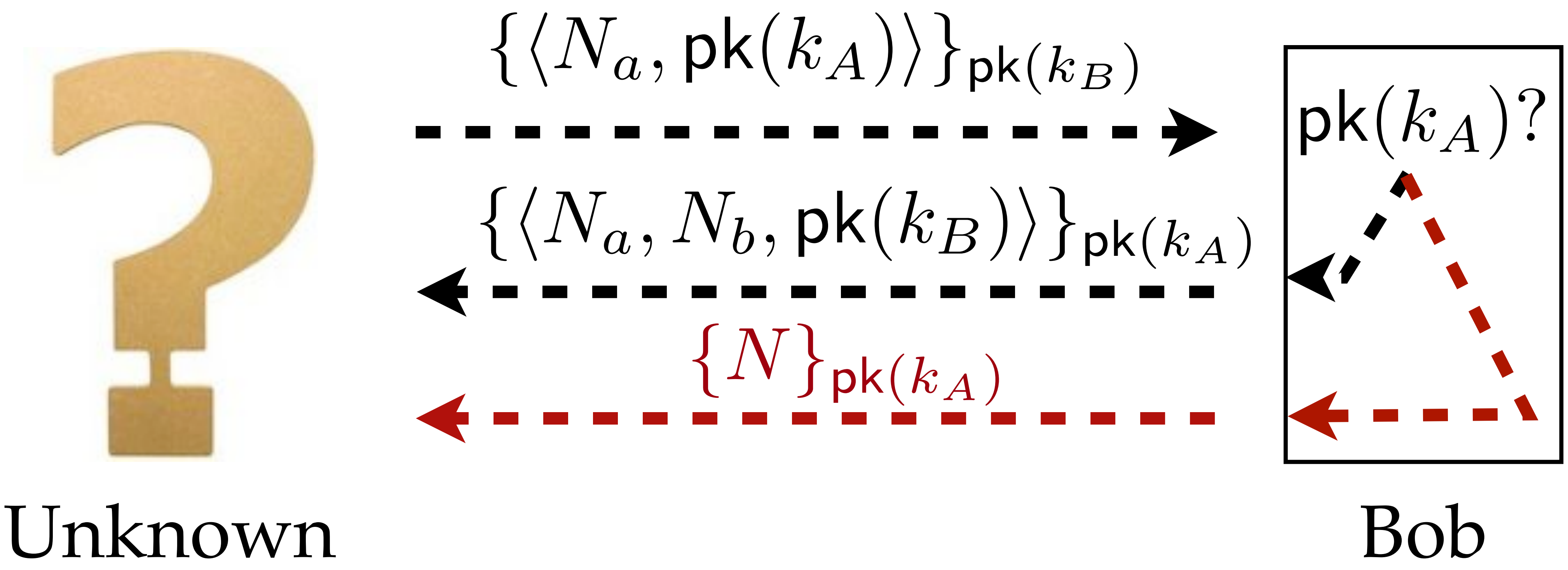
$\{N\}_{\mathsf{pk}(k_A)}$

$\mathsf{pk}(k_A)?$

Bob

Examples

Private authentication protocol



Automatic tools

- For reachability properties

  Avispa, CSP/FDR, ProVerif, Scyther, Maude-NPA, ...

Automatic tools

- For reachability properties

  Avispa, CSP/FDR, ProVerif, Scyther, Maude-NPA, ...

- For equivalence properties

  • **ProVerif:** Bruno Blanchet. *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules.*
  • **SPEC:** Alwen Tiu and Jeremy E. Dawson. *Automating open bisimulation checking for the spi calculus.*
  • **AKiSs:** Ştefan Ciobâcă. *Automated Verification of Security Protocols with Applications to Electronic Voting.*

Automatic tools

- For reachability properties

  Avispa, CSP/FDR, ProVerif, Scyther, Maude-NPA, ...

- For equivalence properties

  • **ProVerif:** Bruno Blanchet. *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules.*
  • **SPEC:** Alwen Tiu and Jeremy E. Dawson. *Automating open bisimulation checking for the spi calculus.*
  • **AKiSs:** Ştefan Ciobâcă. *Automated Verification of Security Protocols with Applications to Electronic Voting.*

> Do not handle private authentication protocol and e-passport protocol

My contributions

- Relations between different notions of equivalences
- Algorithms to prove equivalence
  - ✓Extension of ProVerif
  - ✓New decision procedure for trace equivalence (else branches)
  - ✓New automatic tool : APTE
- Composition result for trace equivalence
  - ✓Application on the e-passport protocol

Outline

1. Proving more equivalence with ProVerif

2. Decision procedure for trace equivalence

3. Composing trace equivalence

Outline

1. Proving more equivalence with ProVerif
2. Decision procedure for trace equivalence
3. Composing trace equivalence

Proverif

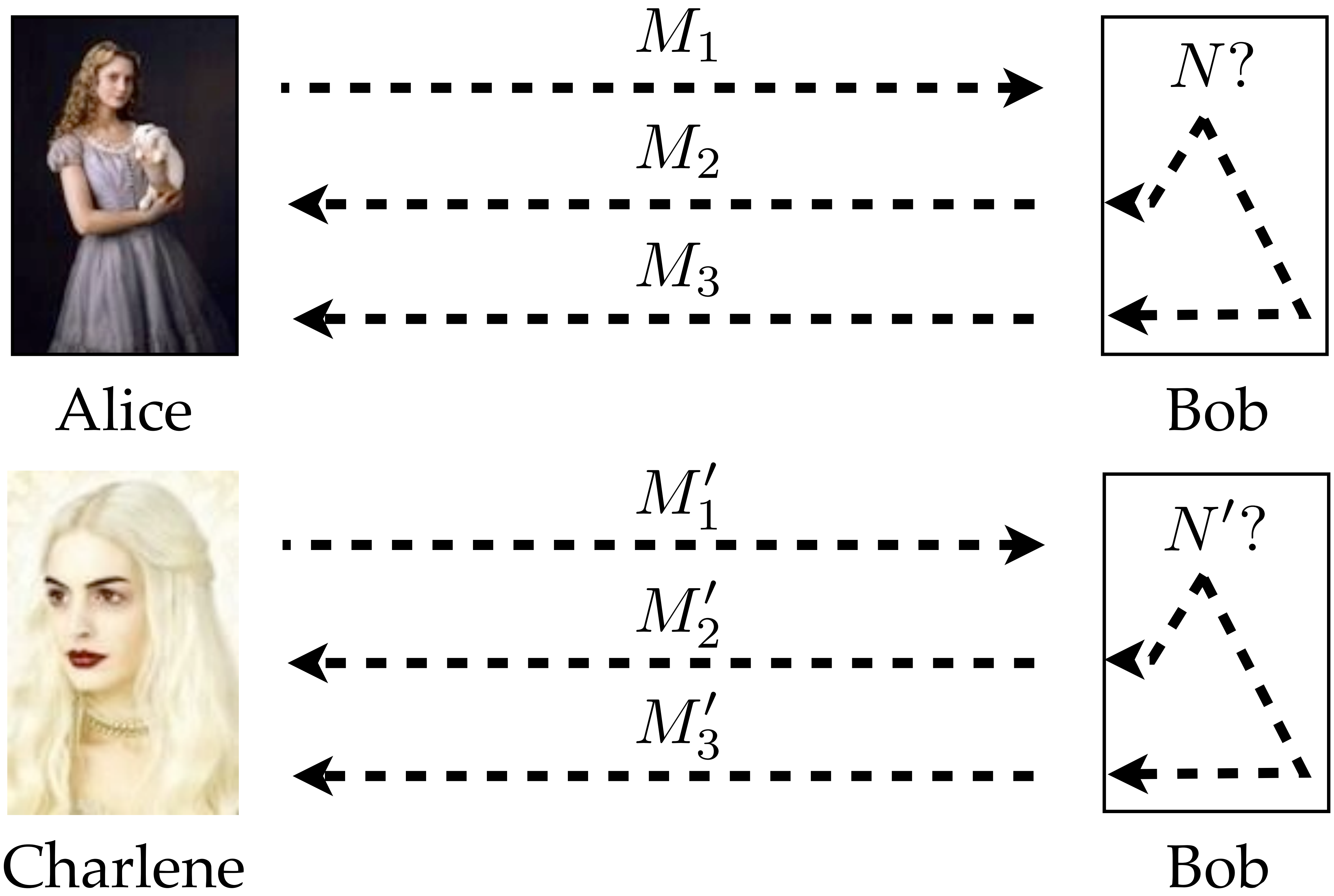
ProVerif was first an analyzer for reachability properties based on Horn clauses.

- Handle reachability and equivalence properties
- Cryptographic primitives described by equational theory and/or rewriting rules
- Handle processes with replication
- Possible false attack
- Does not always terminate

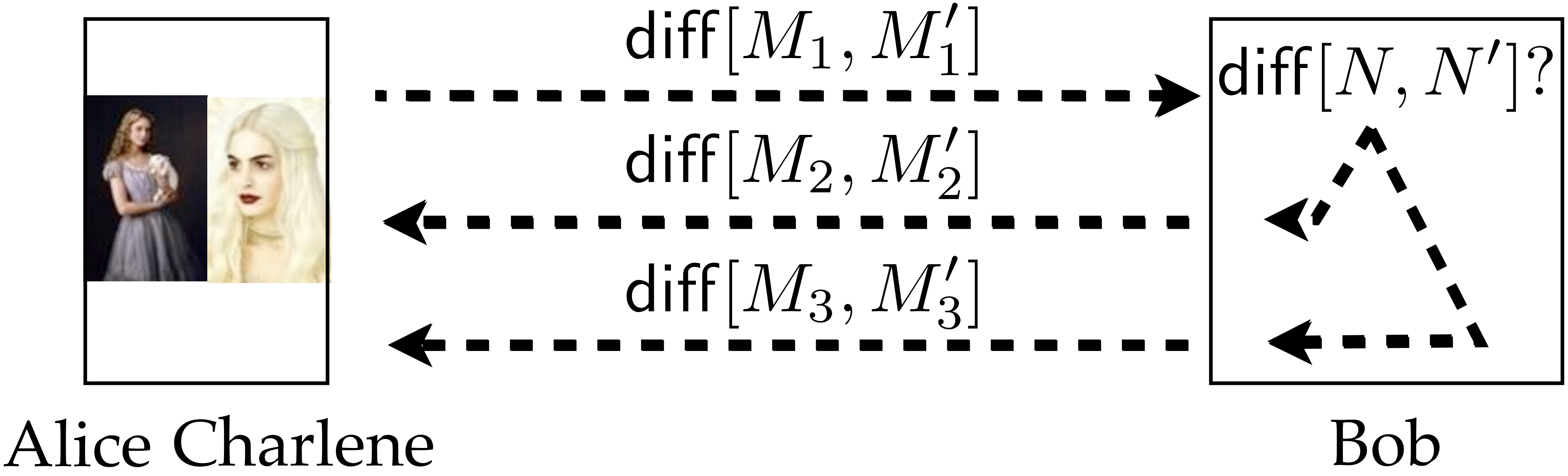
Processes

$$
\begin{aligned}
P, Q := \quad & 0 \\
& \mathsf{in}(c, x); P \\
& \mathsf{out}(c, M); P \\
& P \mid Q \\
& !\, P \\
& \mathsf{new}\ a; P \\
& \mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q
\end{aligned}
$$

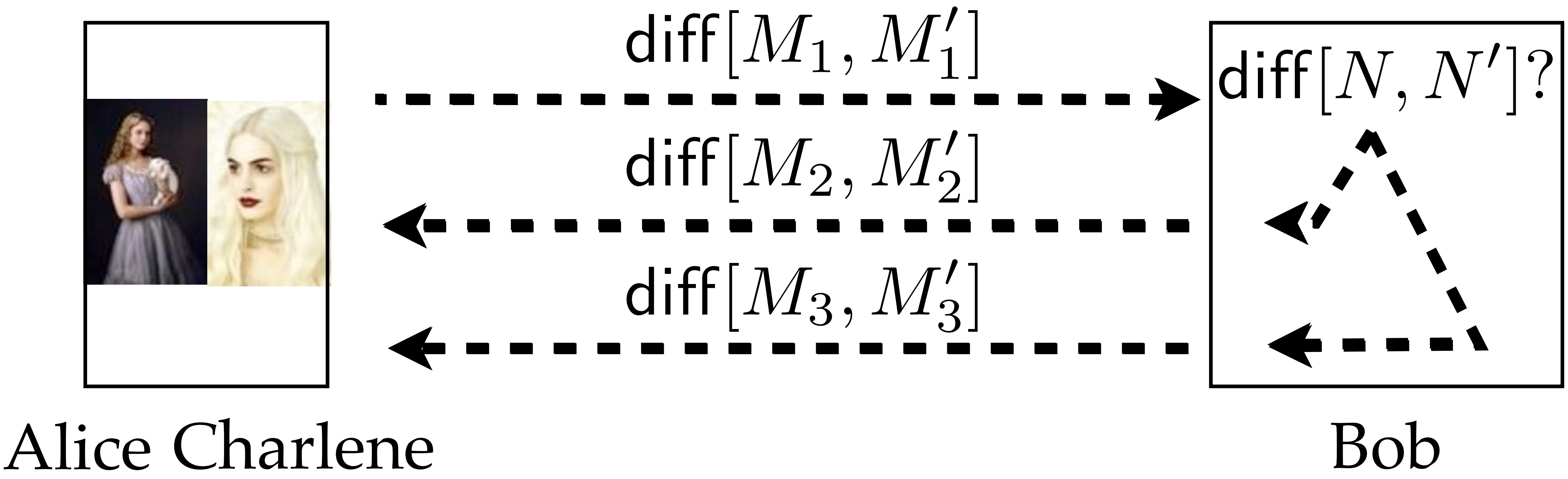
Biprocesses



Biprocesses



Biprocesses



Equivalence too strong: possible false attack

Motivation

The private authentication protocol

Alice

Attacker

Bob

Charlene

Attacker

Bob

Motivation

The private authentication protocol

Alice

Bob

Charlene

Bob

Motivation

The private authentication protocol

Alice

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$

Bob

Charlene

Bob

Motivation

The private authentication protocol

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$

Alice

$\mathsf{pk}(k_A) = y$

Bob

Charlene

Bob

Motivation

The private authentication protocol

Alice

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$

$\mathsf{pk}(k_A) = y$

$\{\langle x, N_b, \mathsf{pk}(k_B)\rangle\}_y$

Bob

Charlene

Bob

Motivation

The private authentication protocol

Alice

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$

$\mathsf{pk}(k_A) = y$

$\{\langle x, N_b, \mathsf{pk}(k_B)\rangle\}_y$

$\{N\}_{\mathsf{pk}(k_A)}$

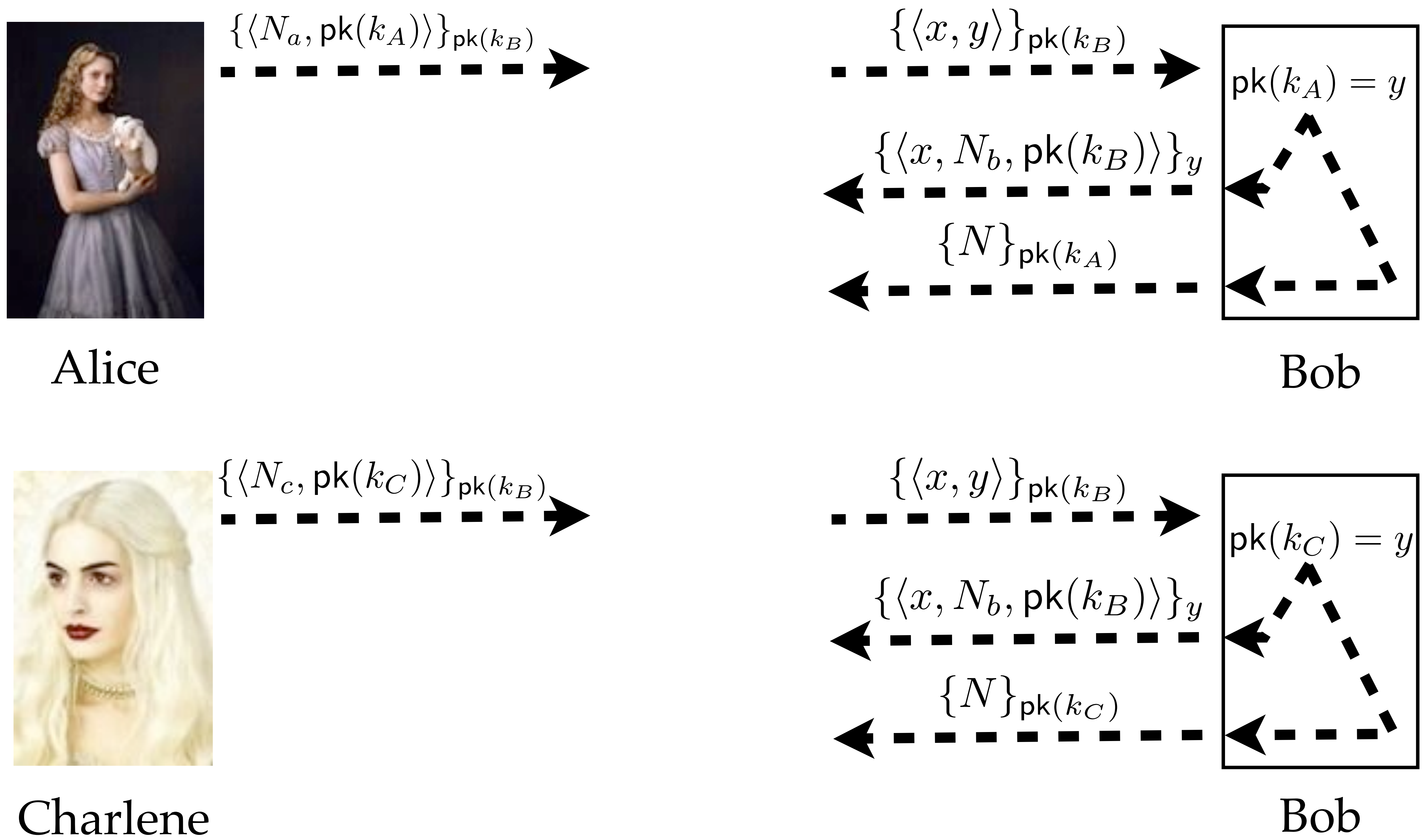
Bob

Charlene

Bob

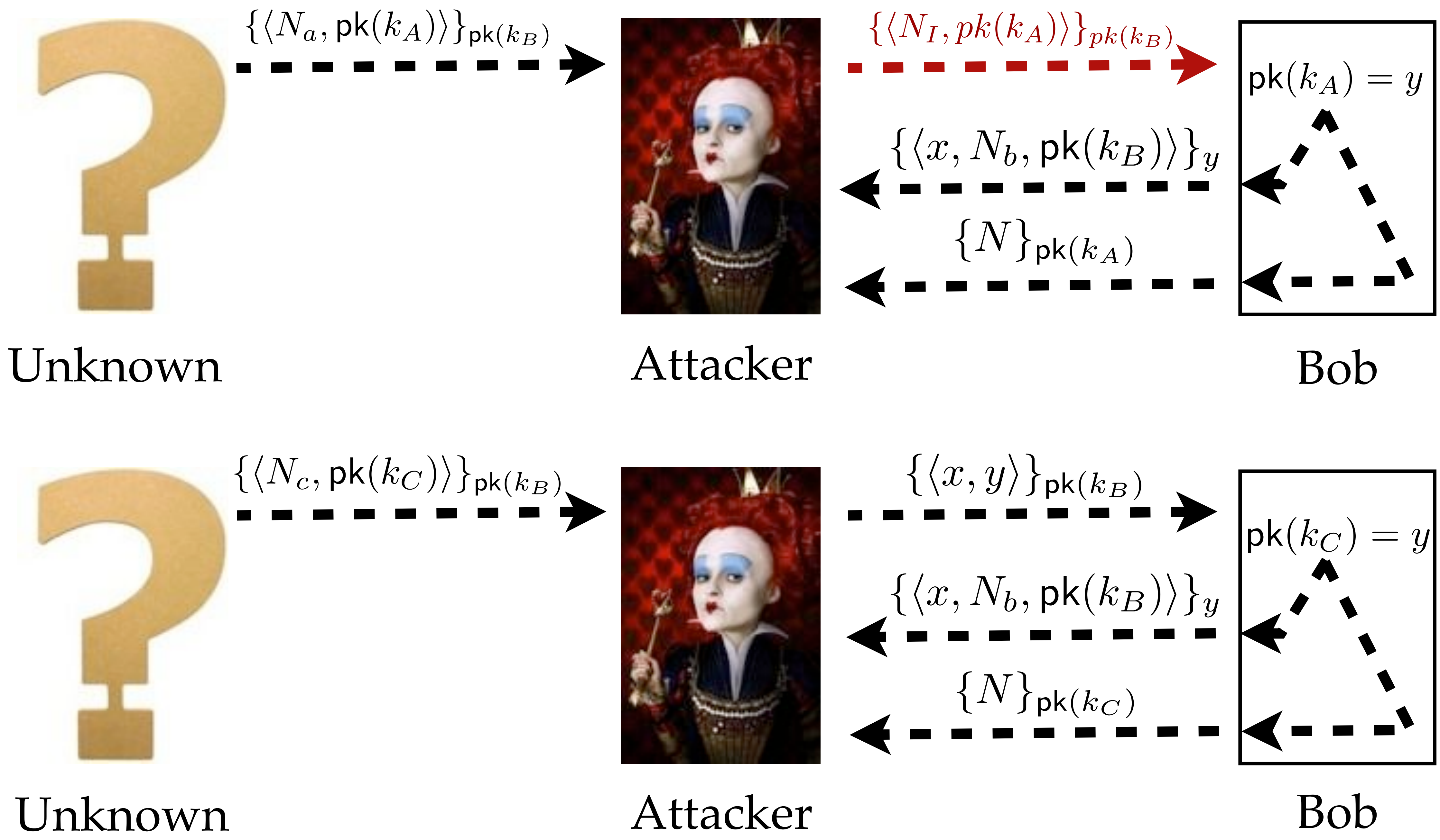
Motivation

The private authentication protocol



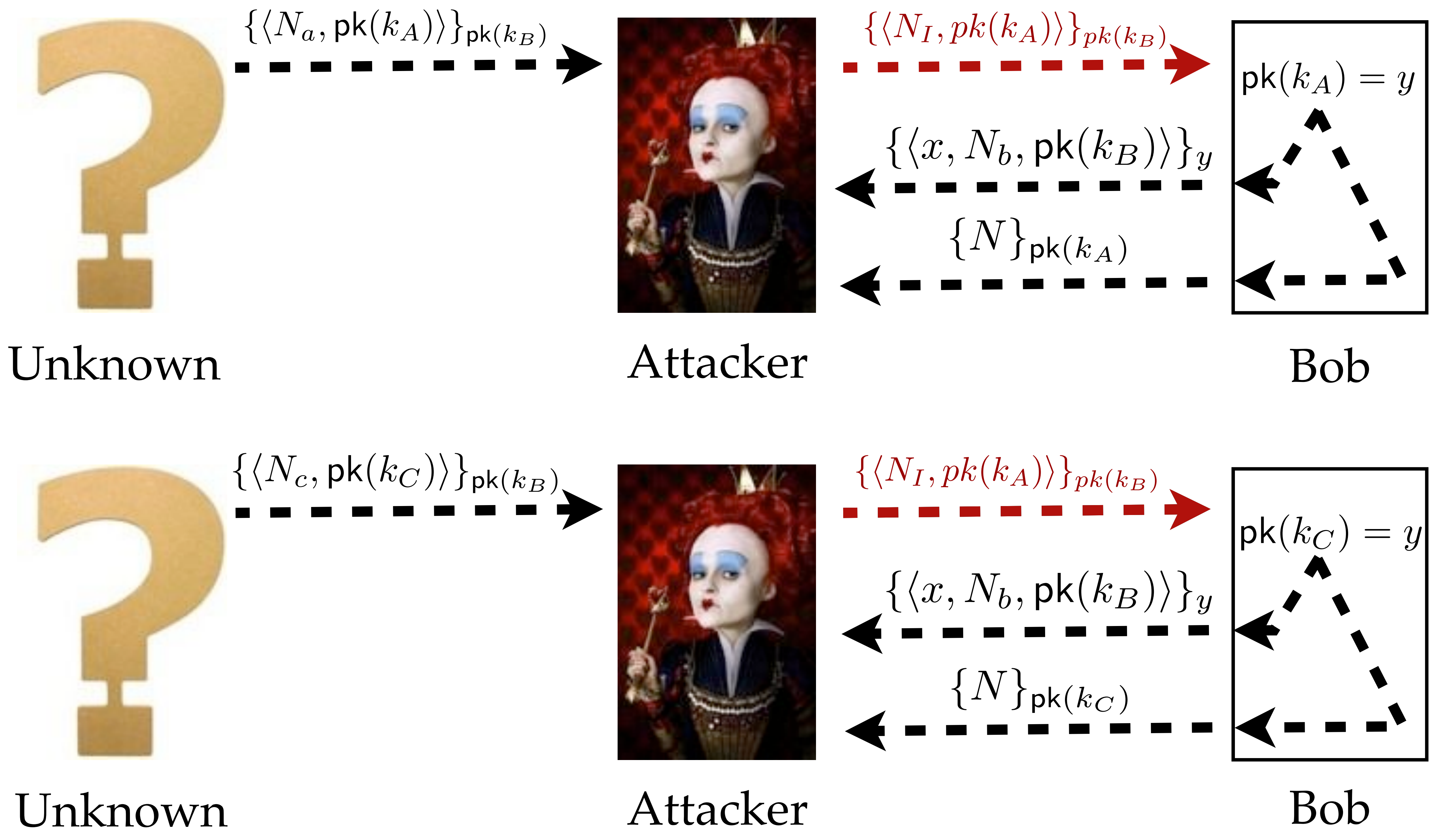
Motivation

The private authentication protocol



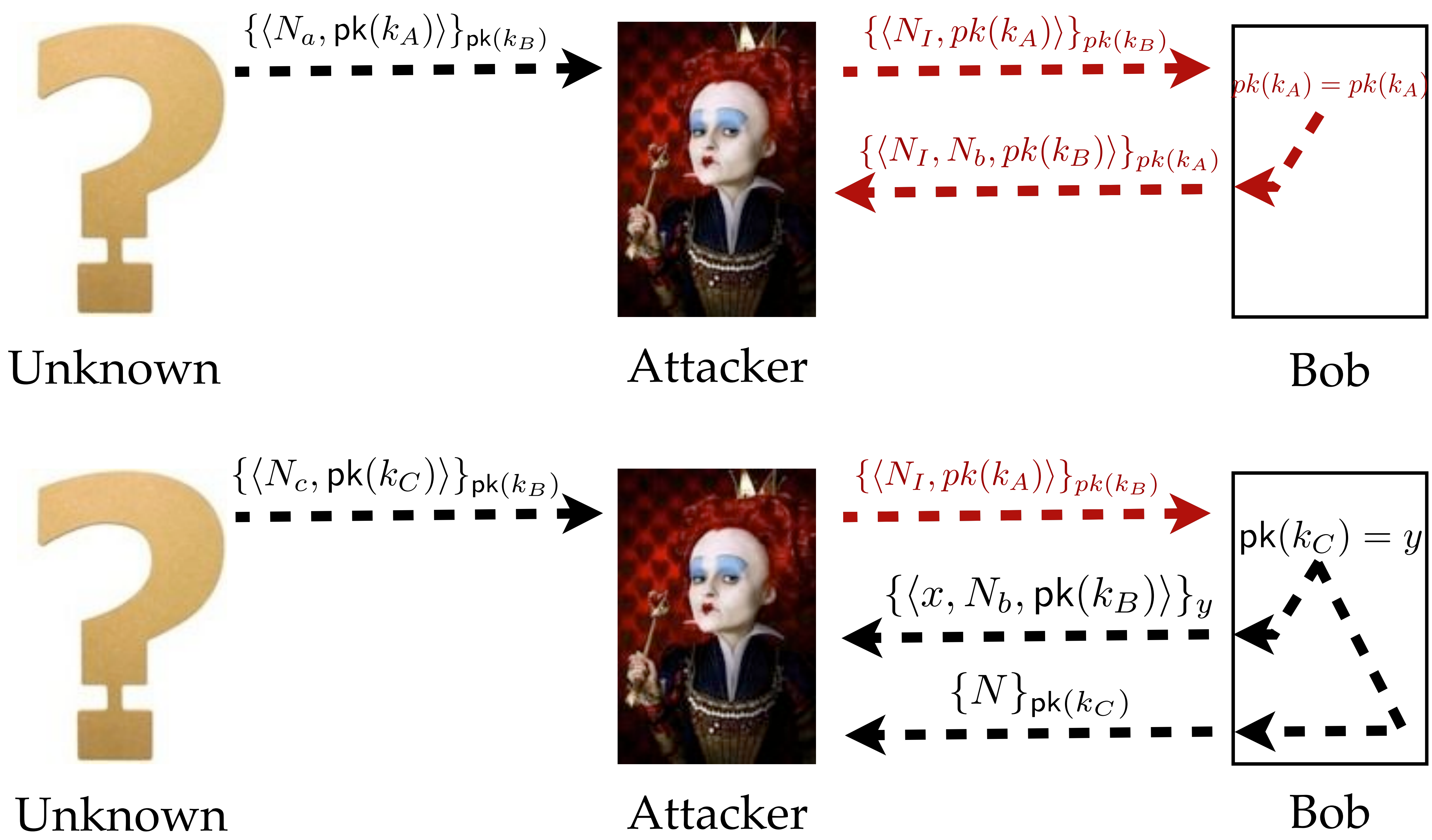
Motivation

The private authentication protocol



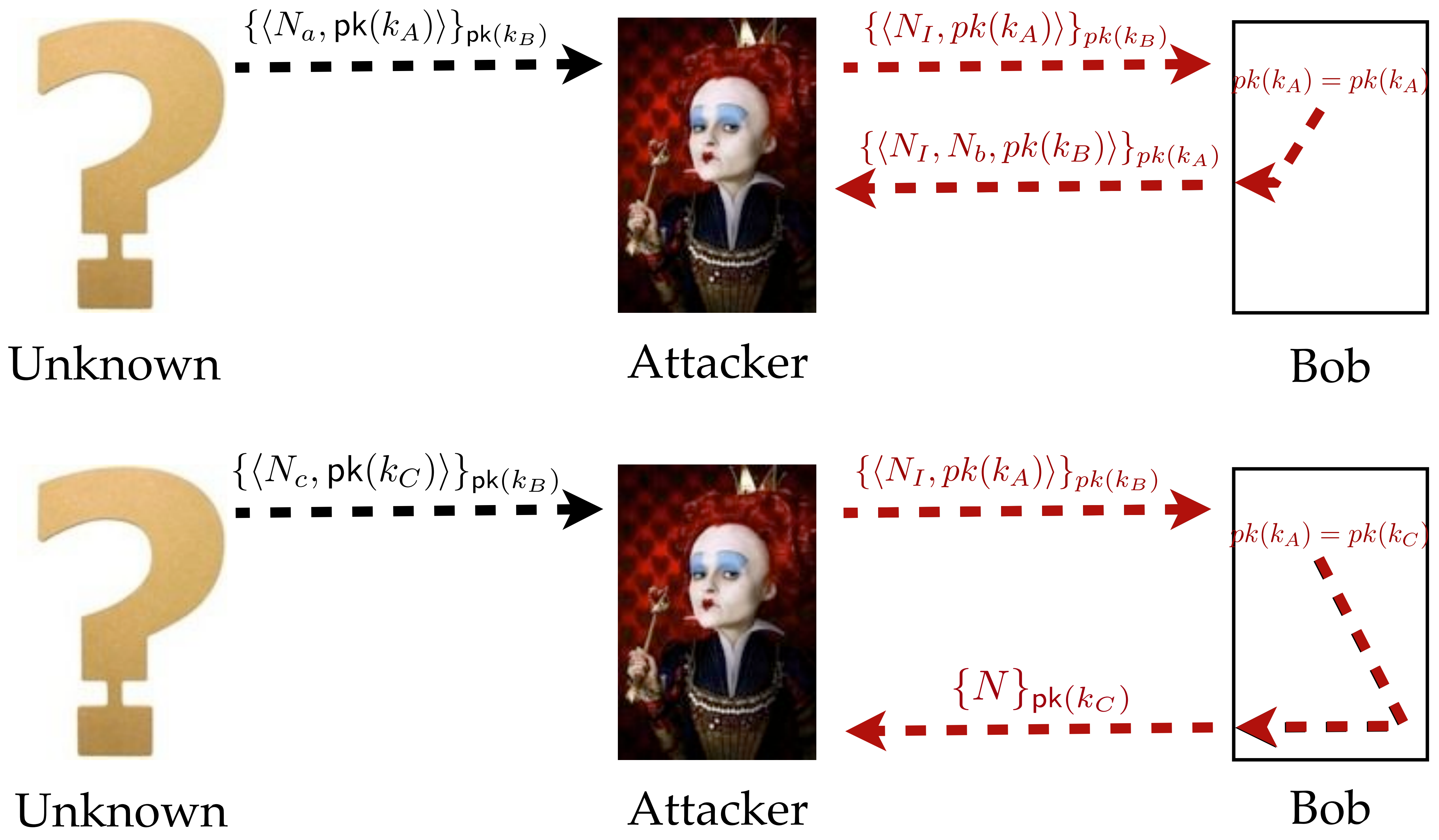
Motivation

The private authentication protocol



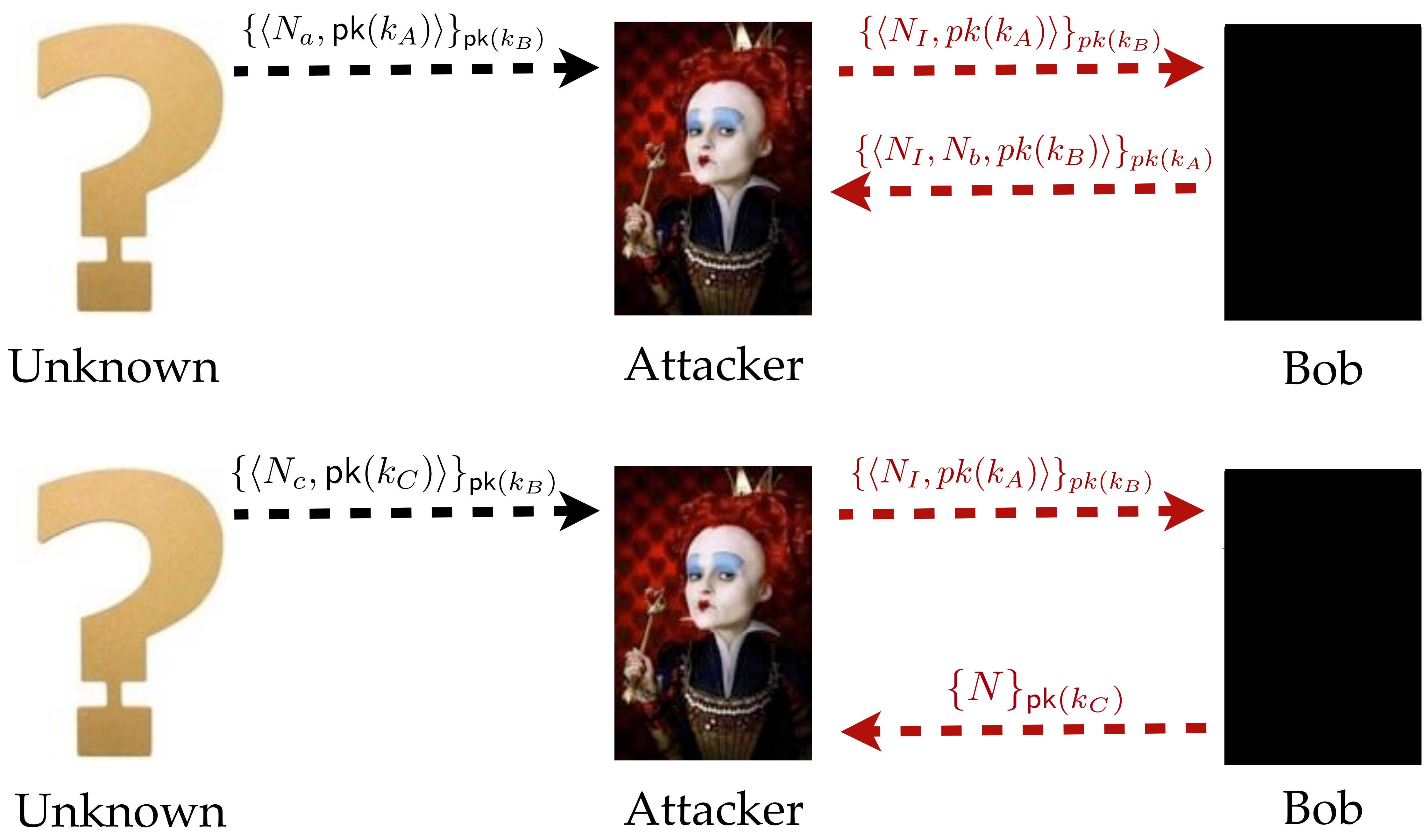
Motivation

The private authentication protocol



Motivation

The private authentication protocol



Contribution

Introduction of destructors with tests between terms

Contribution

Introduction of destructors with tests between terms

The *If-then-else* destructor:

$$\mathsf{ifthenelse}(x, x, z, t) \rightarrow z$$

$$\mathsf{ifthenelse}(x, y, z, t) \rightarrow t \quad \text{with} \quad x \neq y$$

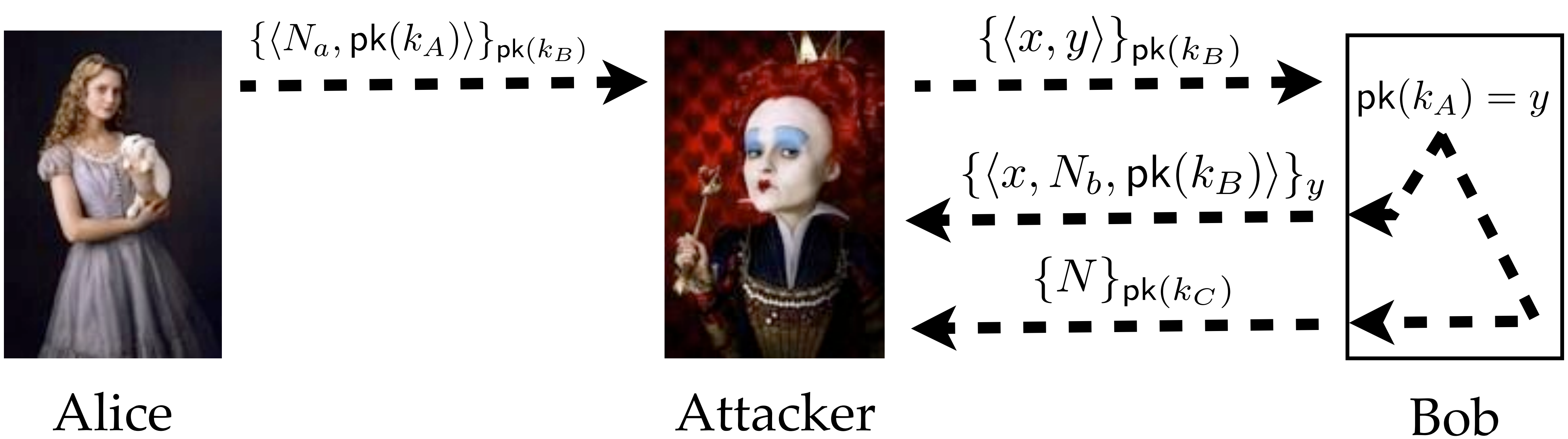
Contribution

Introduction of destructors with tests between terms

The *If-then-else* destructor:

$$\mathsf{ifthenelse}(x, x, z, t) \rightarrow z$$

$$\mathsf{ifthenelse}(x, y, z, t) \rightarrow t \quad \text{with} \quad x \neq y$$



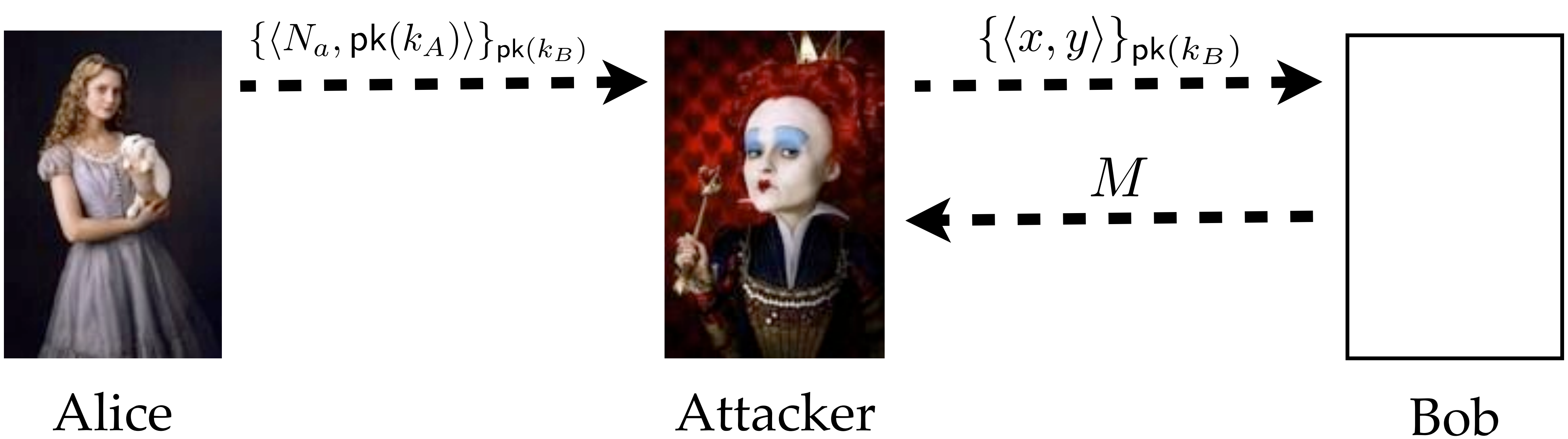
Contribution

Introduction of destructors with tests between terms

The *If-then-else* destructor:

$$\mathsf{ifthenelse}(x, x, z, t) \rightarrow z$$

$$\mathsf{ifthenelse}(x, y, z, t) \rightarrow t \quad \text{with} \quad x \neq y$$



$$M = \mathsf{ifthenelse}(y, \mathsf{pk}(k_A), \{x, N_b, \mathsf{pk}(k_B)\}_y, \{N\}_{\mathsf{pk}(k_A)})$$

Contribution

Introduction of destructors with tests between terms

The *If-then-else* destructor:

$$\mathsf{ifthenelse}(x, x, z, t) \rightarrow z$$

$$\mathsf{ifthenelse}(x, y, z, t) \rightarrow t \quad \text{with} \quad x \neq y$$

Automatic transformation: *simpl*

For all processes P, $simpl(P) \approx P$

Implementation

Beta release:

ProVerif version 1.87beta

http://prosecco.gforge.inria.fr/personal/bblanche/proverif/

Content:

- Rewrite rules with tests
- Automatic transformation of biprocesses
- Equivalence between processes with different control structures

Results:

- Prove anonymity for private authentication protocol (unbounded number of sessions)

Outline

1. Proving more equivalence with ProVerif
2. Decision procedure for trace equivalence
3. Composing trace equivalence

Constraint systems

One constraint system = several traces

Alice

Attacker

Bob

$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I$

Constraint systems

One constraint system = several traces

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

Alice

Attacker

Bob

$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

Constraint systems

One constraint system = several traces

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$

Alice

Attacker

Bob

$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)} \vdash \{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$$

Constraint systems

One constraint system = several traces

$\{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}$

$\{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$

$y = \mathsf{pk}(k_A)$

Alice Attacker Bob

$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)} \vdash \{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$$

$$y = \mathsf{pk}(k_A)$$

Constraint systems

One constraint system = several traces

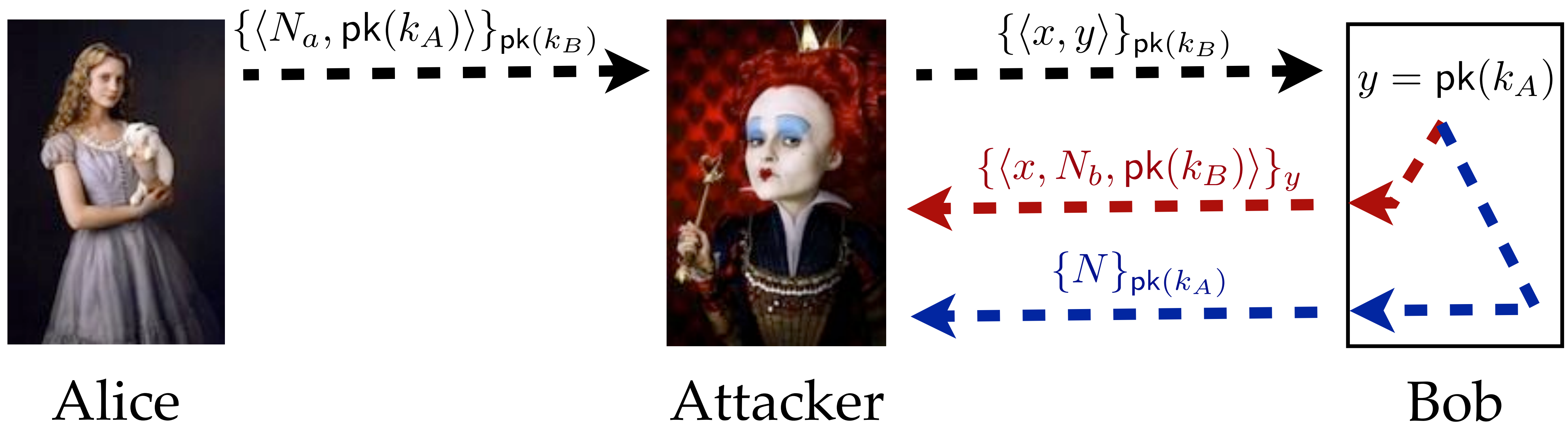
$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)} \vdash \{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$$

$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}, \{\langle x, N_b, \mathsf{pk}(k_B)\rangle\}_y$$

$$y = \mathsf{pk}(k_A)$$

Constraint systems

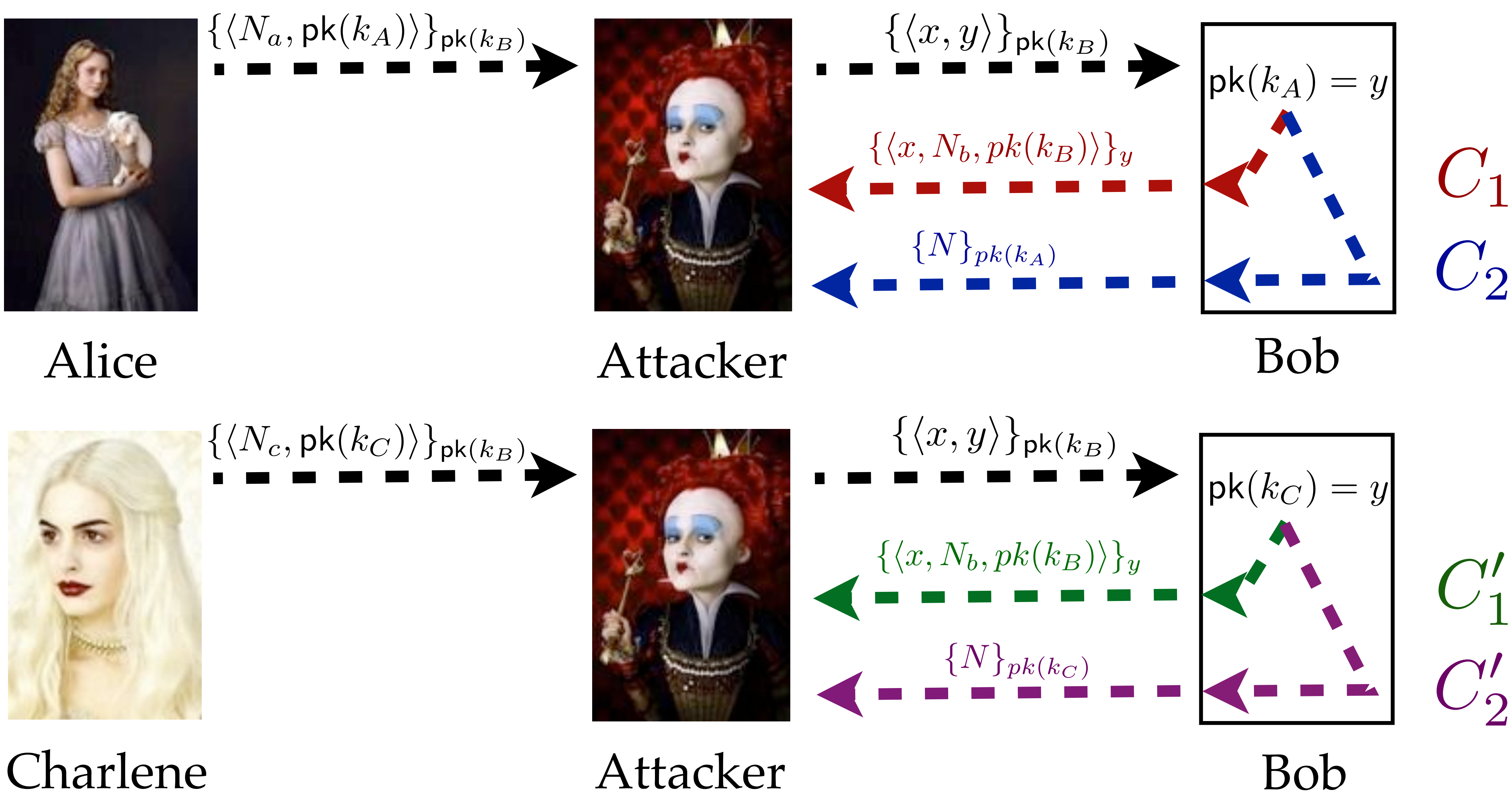
One constraint system = several traces



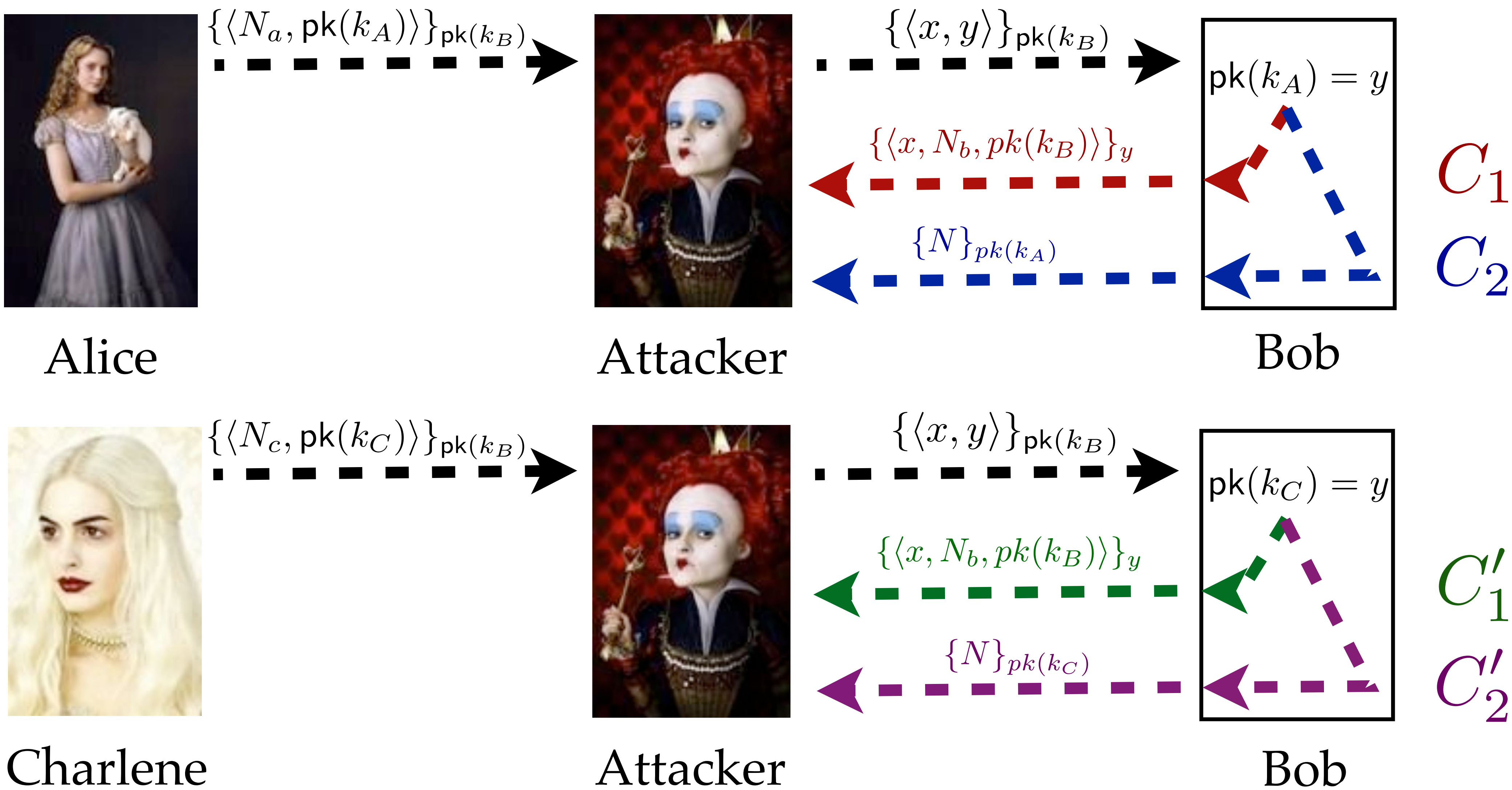
$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)} \vdash \{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$$
$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}, \{\langle x, N_b, \mathsf{pk}(k_B)\rangle\}_y$$
$$y = \mathsf{pk}(k_A)$$

$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)} \vdash \{\langle x, y\rangle\}_{\mathsf{pk}(k_B)}$$
$$\mathsf{pk}(k_A), \mathsf{pk}(k_B), \mathsf{pk}(k_C), N_I, \{\langle N_a, \mathsf{pk}(k_A)\rangle\}_{\mathsf{pk}(k_B)}, \{N\}_{\mathsf{pk}(k_A)}$$
$$y \neq \mathsf{pk}(k_A)$$

Sets of constraint systems

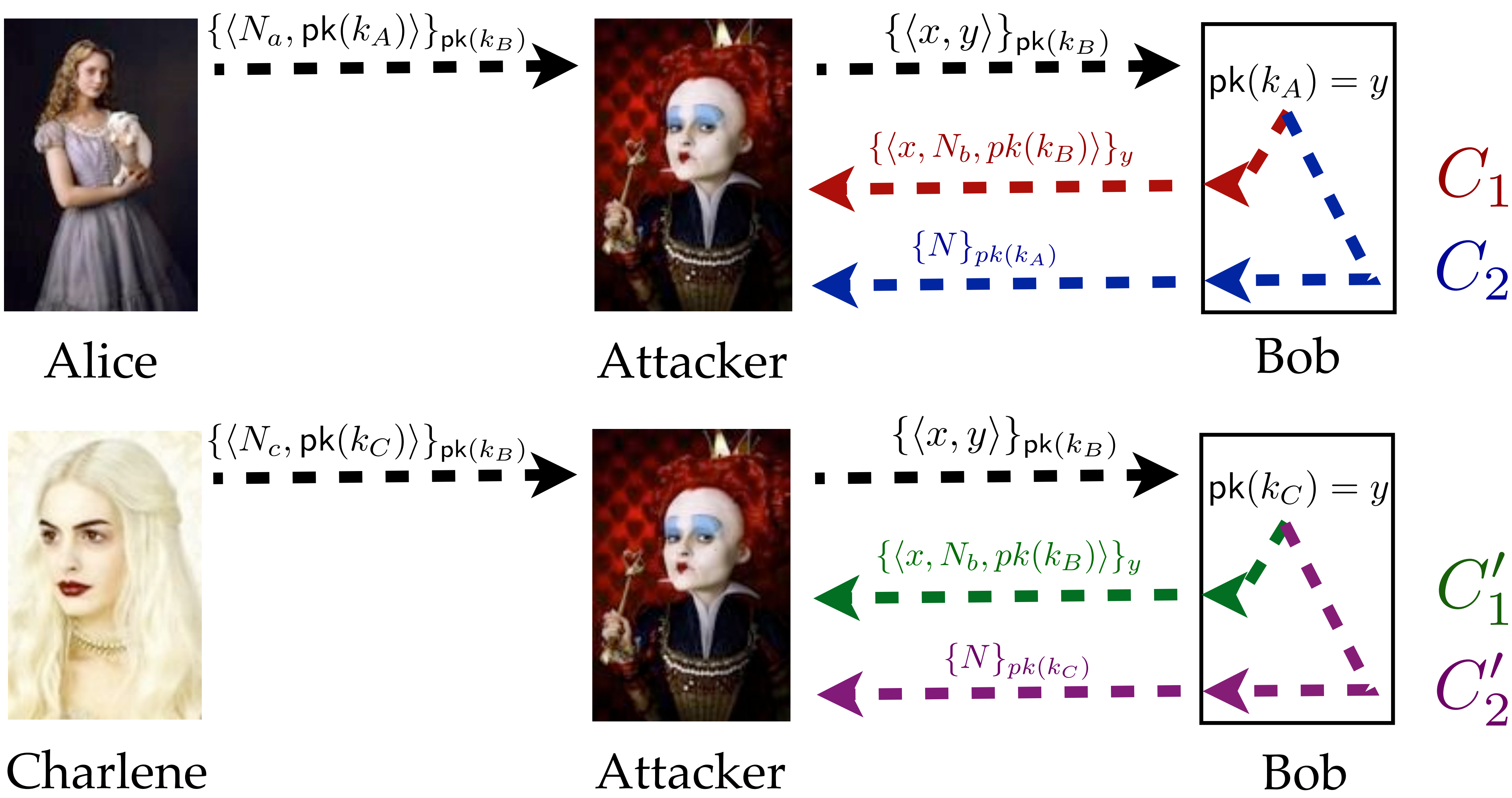


Sets of constraint systems



$$\{C_1;\ C_2\} \approx \{C'_1;\ C'_2\}$$

Sets of constraint systems



Symbolic equivalence between sets of constraint systems

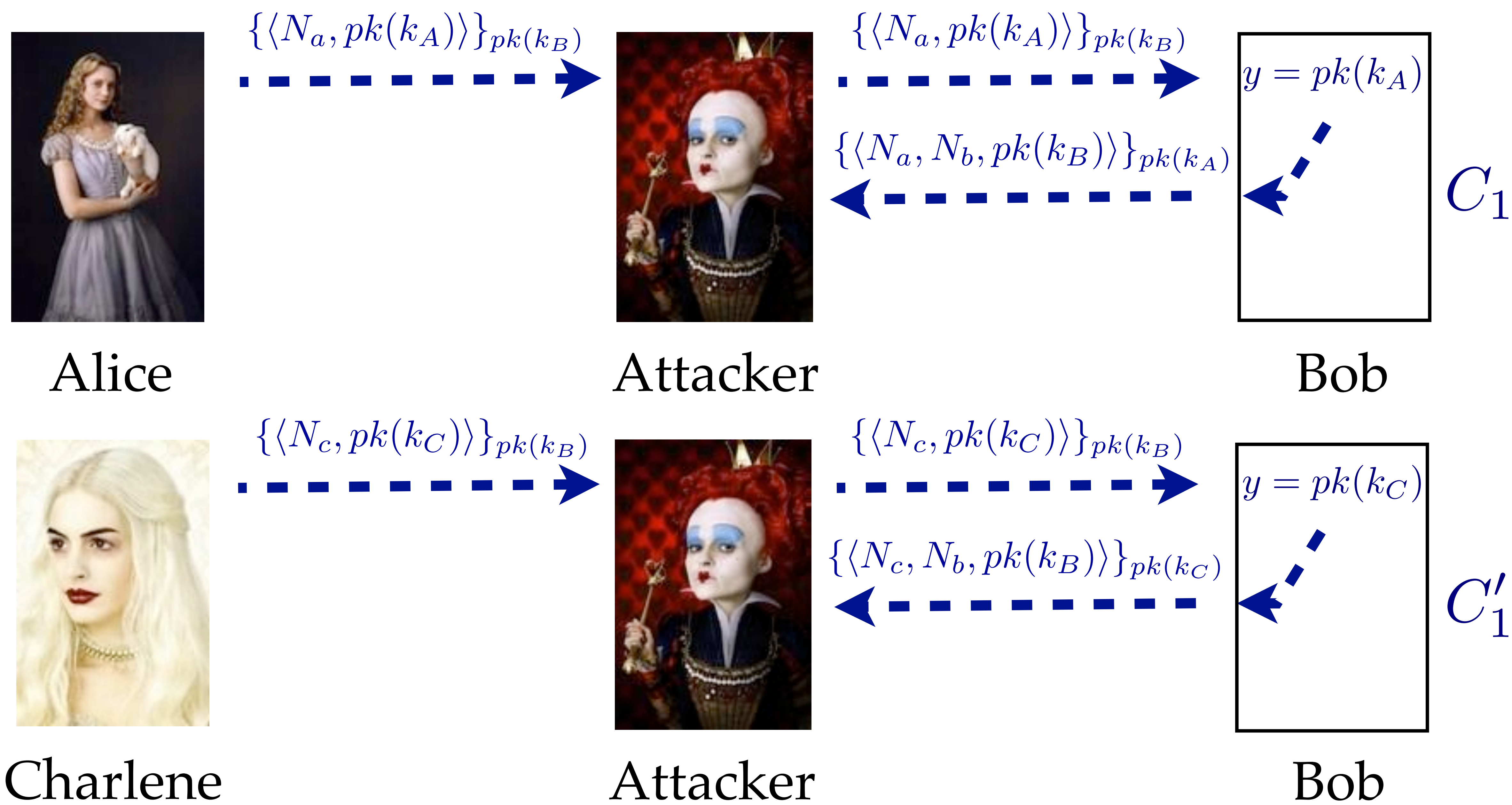
Previous works

1. M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Phd thesis (2007)
2. Y. Chevalier and M. Rusinowitch. *Decidability of equivalence of symbolic derivations*. JAR (2012)
3. A. Tiu and J. E. Dawson. *Automating open bisimulation checking for the spi calculus*. CSF (2010)

- Do not handle set of constraint systems
- Do not handle inequations
- Do not handle non-deterministic processes

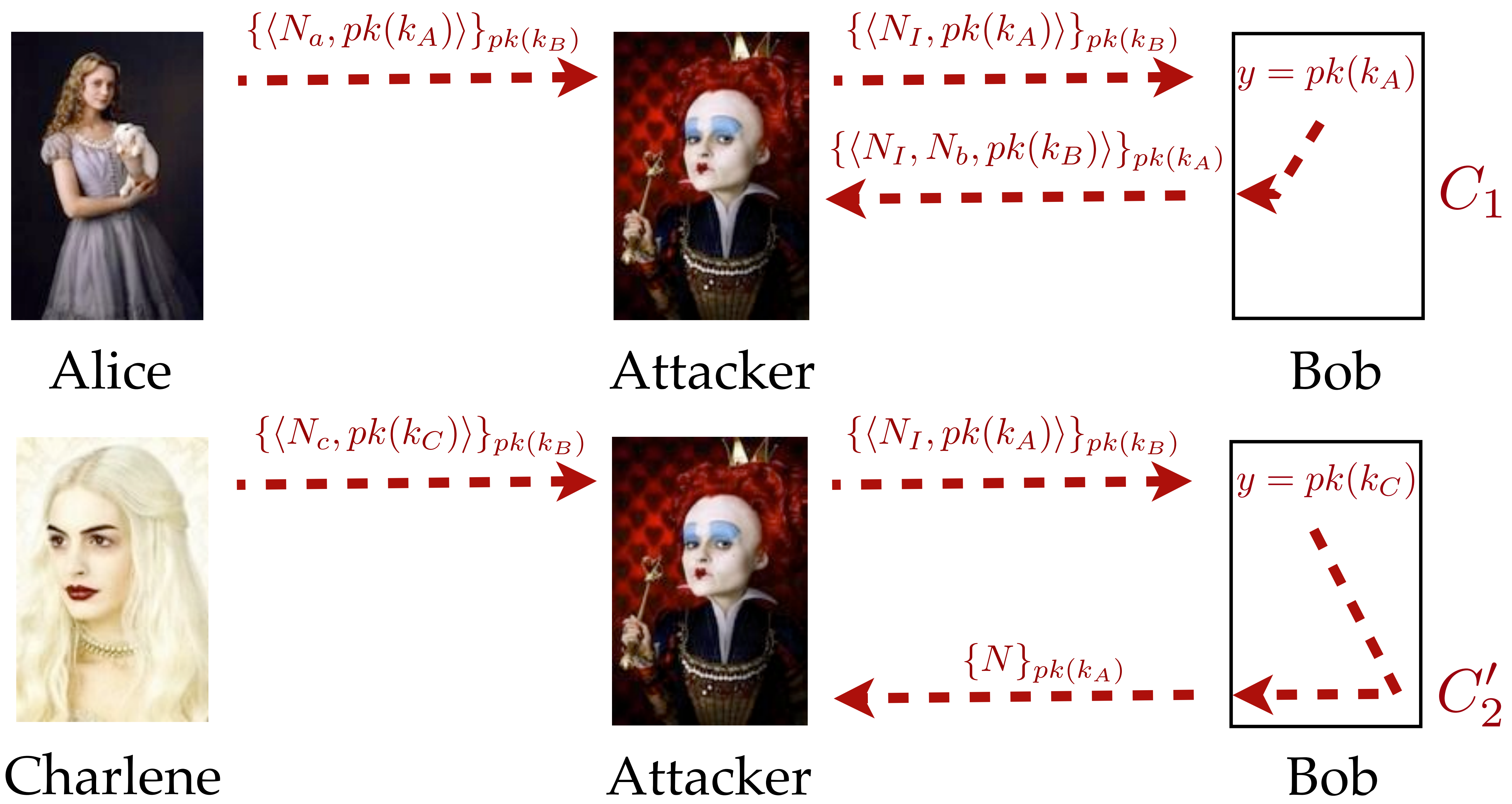
Sets of constraint systems

Why are they necessary ?



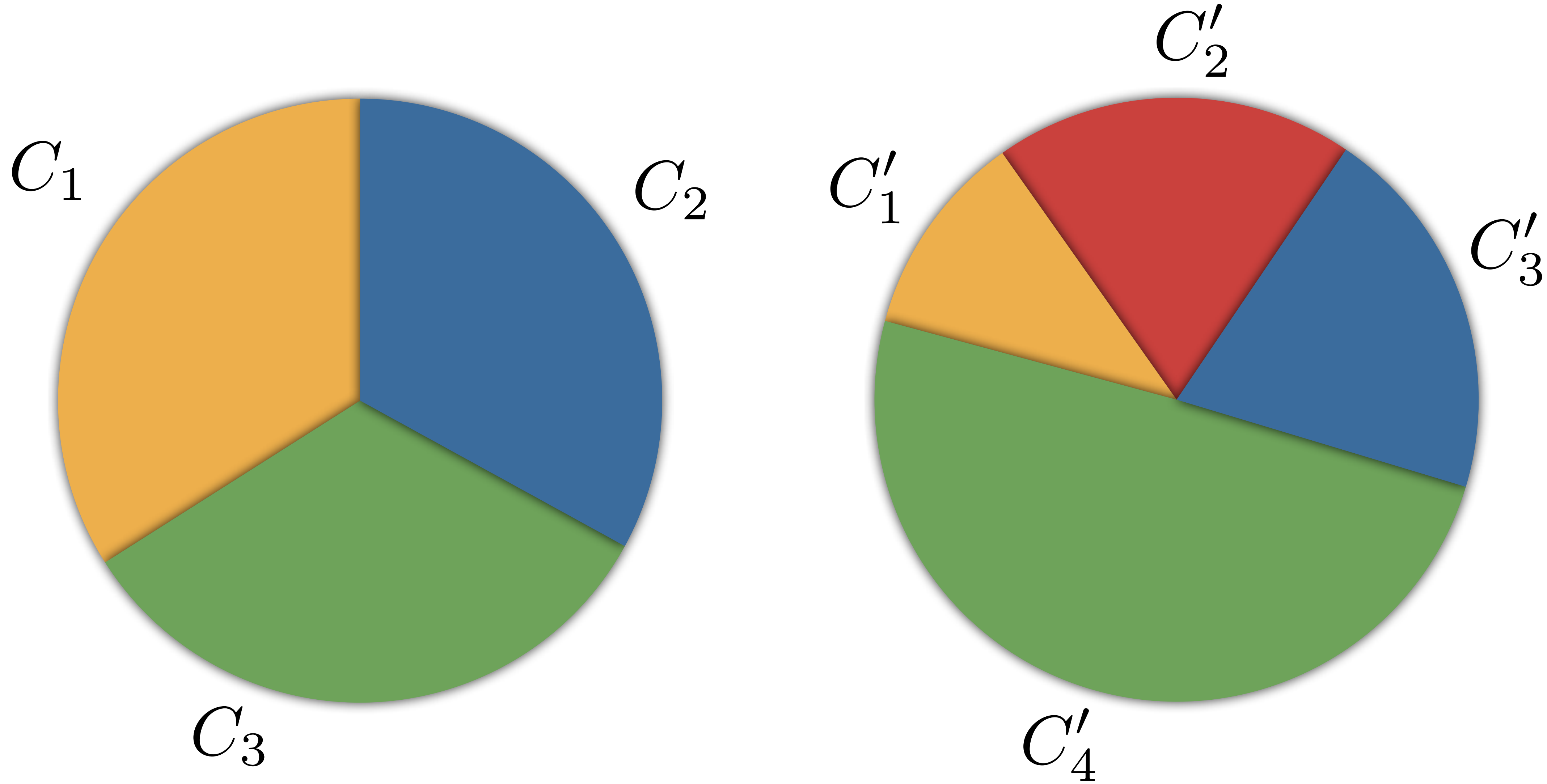
Sets of constraint systems

Why are they necessary ?



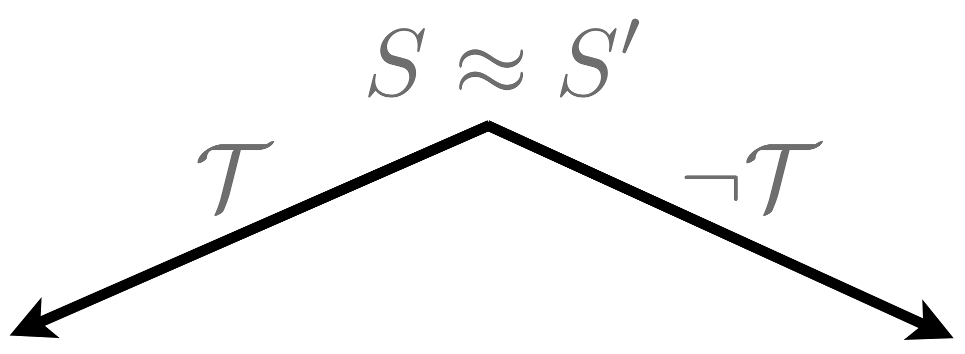
Sets of constraint systems

Why are they necessary ?

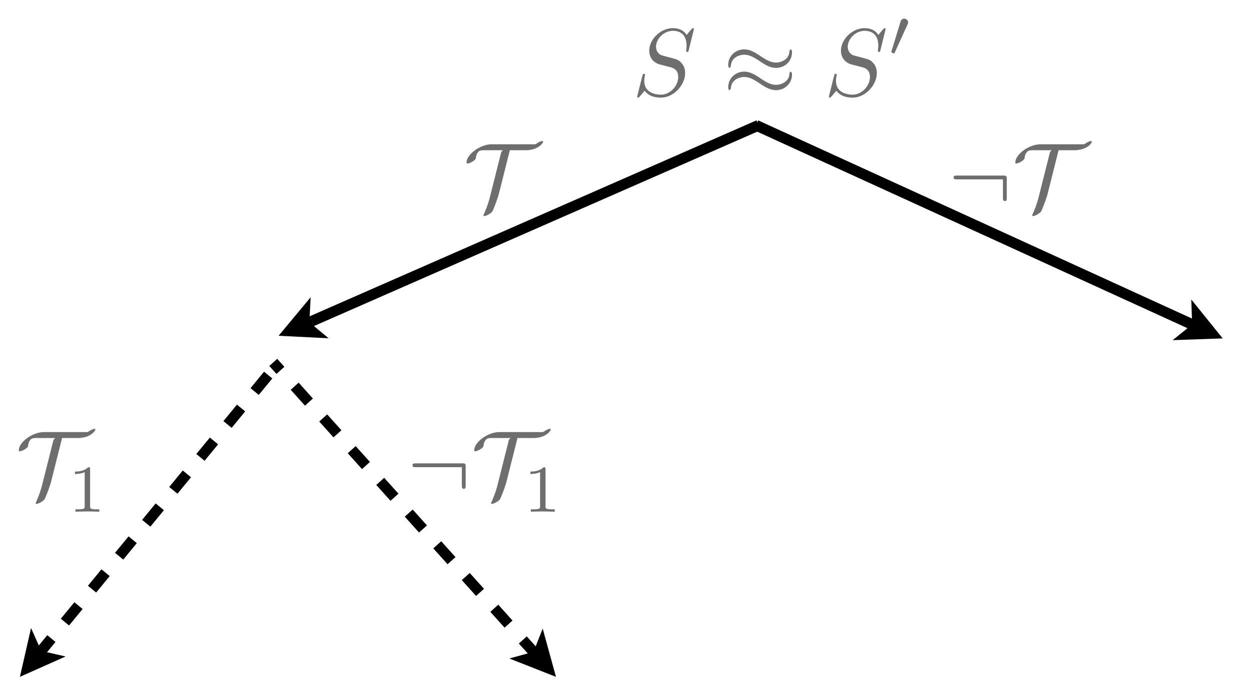


$S = \{C_1; C_2; C_3\}$ $\qquad$ $S' = \{C'_1; C'_2; C'_3; C'_4\}$

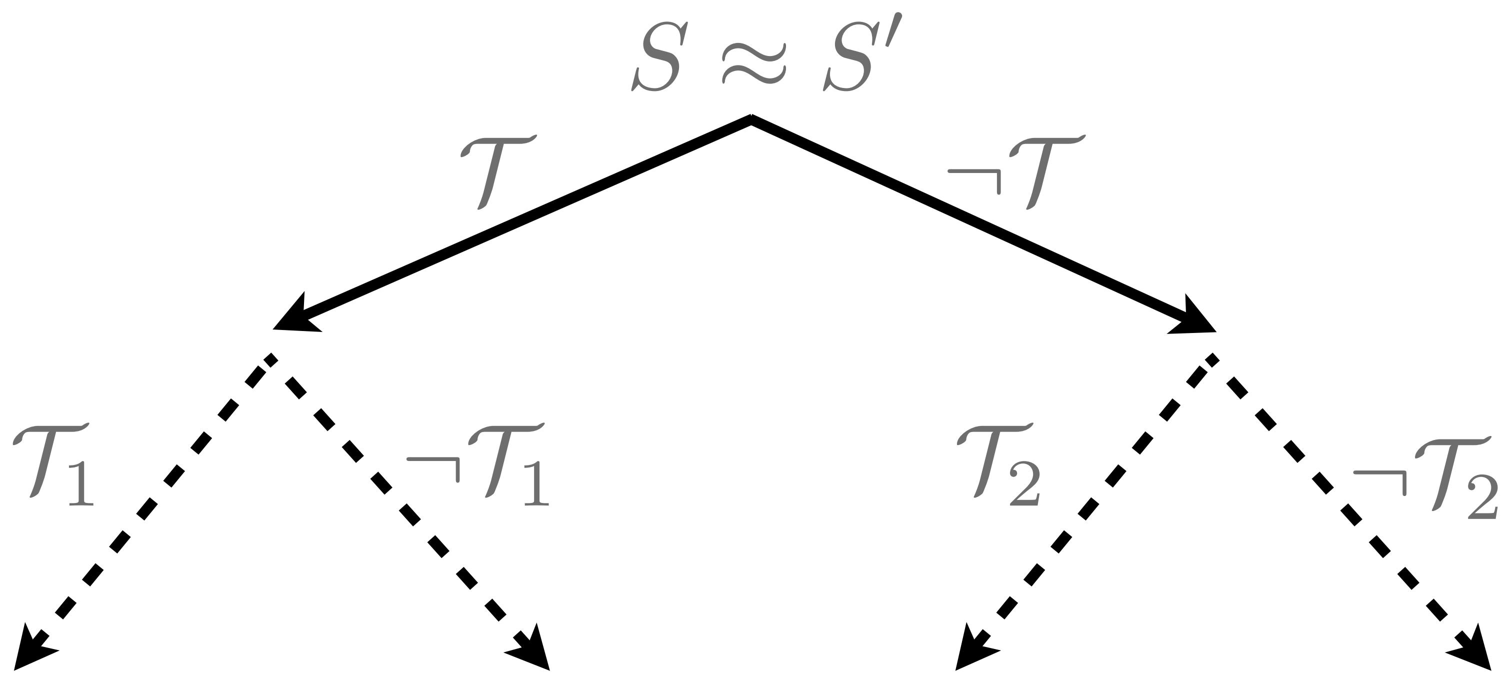
The Algorithm



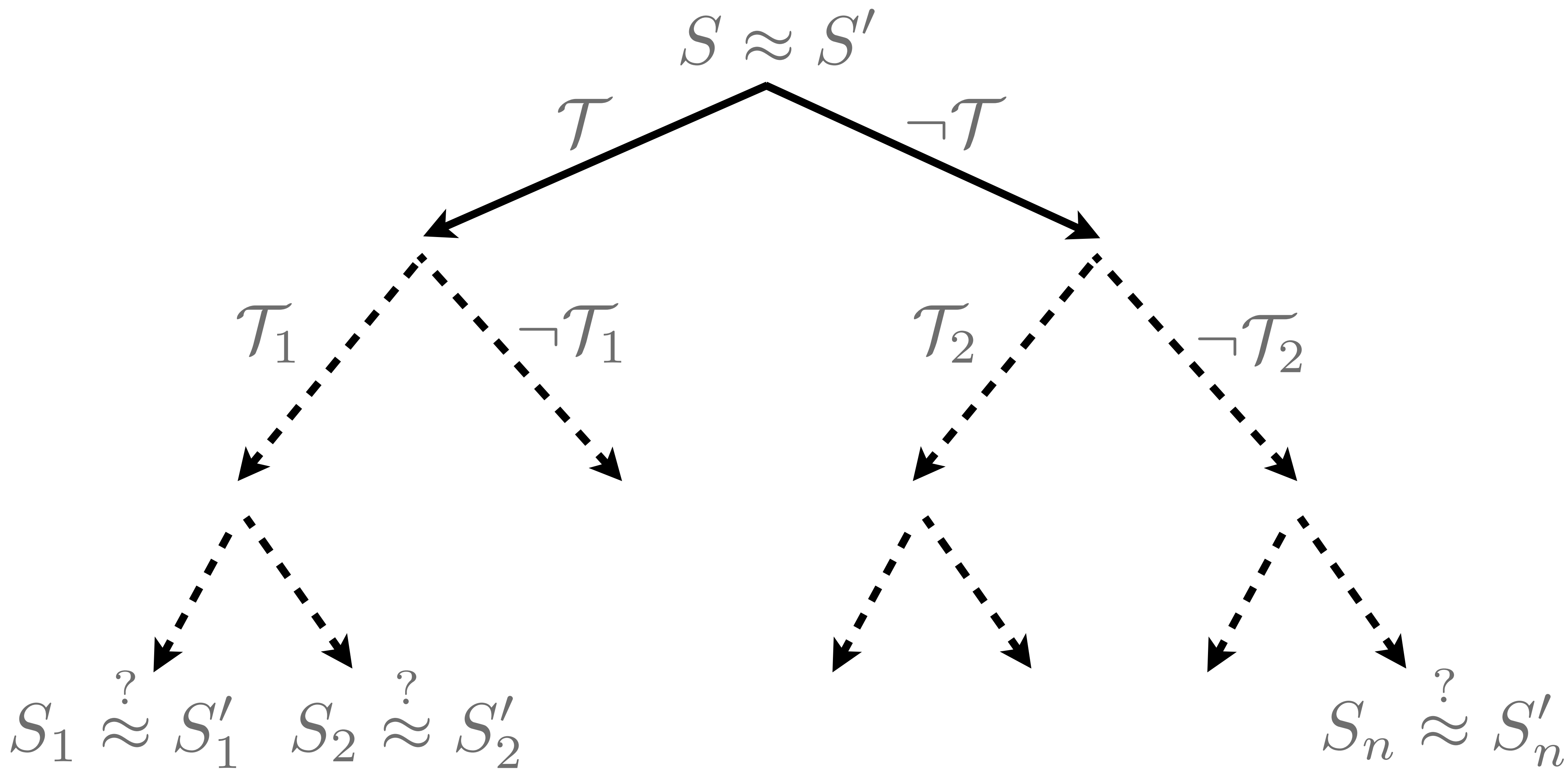
The Algorithm



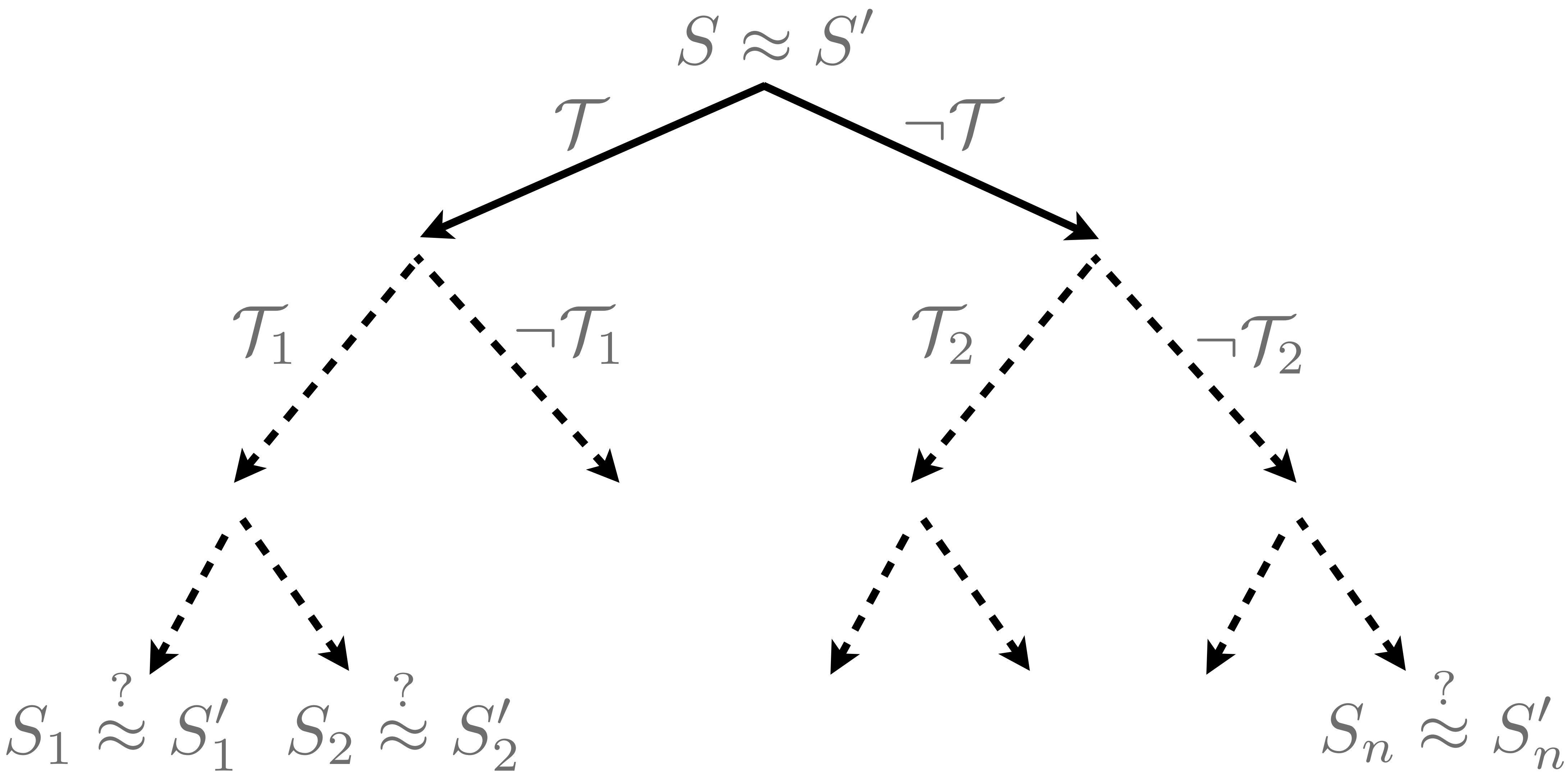
The Algorithm



The Algorithm



The Algorithm



The symbolic equivalence is syntactically decided on each leaf

The Algorithm

Example of a rule

$$\begin{cases} \dots \\ T \vdash_X f(u_1, u_2) \\ \dots \end{cases}$$

The Algorithm

Example of a rule

$$\text{Test } \mathcal{T} = \exists X_1, X_2 \text{ s.t. } X = f(X_1, X_2)$$

$$\begin{cases} \dots \\ T \vdash_X f(u_1, u_2) \\ \dots \end{cases}$$

The Algorithm

Example of a rule

Test $\mathcal{T} = \exists X_1, X_2 \text{ s.t. } X = f(X_1, X_2)$

$$\begin{cases} \ldots \\ T \vdash_X f(u_1, u_2) \\ \ldots \end{cases}$$

$$\xrightarrow{\mathcal{T}}$$

$$\begin{cases} \ldots \\ T \vdash_{X_1} u_1 \\ T \vdash_{X_2} u_2 \\ X = f(X_1, X_2) \\ \ldots \end{cases}$$

The Algorithm

Example of a rule

Test $\mathcal{T} = \exists X_1, X_2 \text{ s.t. } X = f(X_1, X_2)$

$$\begin{cases} \ldots \\ T \vdash_X f(u_1, u_2) \\ \ldots \end{cases}$$

$\mathcal{T}$ $\neg\mathcal{T}$

$$\begin{cases} \ldots \\ T \vdash_{X_1} u_1 \\ T \vdash_{X_2} u_2 \\ X = f(X_1, X_2) \\ \ldots \end{cases} \qquad \begin{cases} \ldots \\ T \vdash_X f(u_1, u_2) \\ \mathsf{Top}(X) \neq f \\ \ldots \end{cases}$$

Termination difficulties

$$\begin{cases} T \vdash_X x \\ T \vdash_Y f(x, y) \end{cases}$$

$$\begin{cases} T' \vdash_X f(x', y') \\ T' \vdash_Y x' \end{cases}$$

Termination difficulties

$$\begin{cases} T \vdash_X x \\ T \vdash_Y f(x, y) \end{cases} \qquad \begin{cases} T' \vdash_X f(x', y') \\ T' \vdash_Y x' \end{cases}$$

$$\Big\downarrow \; X = f(X_1, X_2)$$

Termination difficulties

$$\begin{cases} T \vdash_X x \\ T \vdash_Y f(x, y) \end{cases} \qquad \begin{cases} T' \vdash_X f(x', y') \\ T' \vdash_Y x' \end{cases}$$

$$\Big\downarrow \; X = f(X_1, X_2)$$

$$\begin{cases} T \vdash_{X_1} x_1 \\ T \vdash_{X_2} x_2 \\ T \vdash_Y f(f(x_1, x_2), y) \end{cases} \qquad \begin{cases} T' \vdash_{X_1} x' \\ T' \vdash_{X_2} y' \\ T' \vdash_Y x' \end{cases}$$

Termination difficulties

$$\begin{cases} T \vdash_X x \\ T \vdash_Y f(x, y) \end{cases} \qquad \begin{cases} T' \vdash_X f(x', y') \\ T' \vdash_Y x' \end{cases}$$

$$\Big\downarrow \; X = f(X_1, X_2)$$

$$\begin{cases} T \vdash_{X_1} x_1 \\ T \vdash_{X_2} x_2 \\ T \vdash_Y f(f(x_1, x_2), y) \end{cases} \qquad \begin{cases} T' \vdash_{X_1} x' \\ T' \vdash_{X_2} y' \\ T' \vdash_Y x' \end{cases}$$

$$\Big\downarrow \; Y = f(Y_1, Y_2)$$

Termination difficulties

$$\begin{cases} T \vdash_X x \\ T \vdash_Y f(x, y) \end{cases} \qquad \begin{cases} T' \vdash_X f(x', y') \\ T' \vdash_Y x' \end{cases}$$

$$\downarrow \; X = f(X_1, X_2)$$

$$\begin{cases} T \vdash_{X_1} x_1 \\ T \vdash_{X_2} x_2 \\ T \vdash_Y f(f(x_1, x_2), y) \end{cases} \qquad \begin{cases} T' \vdash_{X_1} x' \\ T' \vdash_{X_2} y' \\ T' \vdash_Y x' \end{cases}$$

$$\downarrow \; Y = f(Y_1, Y_2)$$

$$\begin{cases} T \vdash_{X_1} x_1 \\ T \vdash_{X_2} x_2 \\ T \vdash_{Y_1} f(x_1, x_2) \\ T \vdash_{Y_2} y \end{cases} \qquad \begin{cases} T' \vdash_{X_1} f(x'_1, x'_2) \\ T' \vdash_{X_2} y' \\ T' \vdash_{Y_1} x'_1 \\ T' \vdash_{Y_2} x'_2 \end{cases}$$

Termination difficulties

$$\begin{cases} T \vdash_X x \\ T \vdash_Y f(x, y) \end{cases} \qquad \begin{cases} T' \vdash_X f(x', y') \\ T' \vdash_Y x' \end{cases}$$

$$\Big\downarrow \; X = f(X_1, X_2)$$

$$\begin{cases} T \vdash_{X_1} x_1 \\ T \vdash_{X_2} x_2 \\ T \vdash_Y f(f(x_1, x_2), y) \end{cases} \qquad \begin{cases} T' \vdash_{X_1} x' \\ T' \vdash_{X_2} y' \\ T' \vdash_Y x' \end{cases}$$

$$\Big\downarrow \; Y = f(Y_1, Y_2)$$

$$\begin{cases} T \vdash_{X_1} x_1 \\ T \vdash_{X_2} x_2 \\ T \vdash_{Y_1} f(x_1, x_2) \\ T \vdash_{Y_2} y \end{cases} \qquad \begin{cases} T' \vdash_{X_1} f(x'_1, x'_2) \\ T' \vdash_{X_2} y' \\ T' \vdash_{Y_1} x'_1 \\ T' \vdash_{Y_2} x'_2 \end{cases}$$

Results

Decision procedure for proving trace equivalence for bounded number of sessions.

Class of accepted processes:

- Non-deterministic (e.g. private channel) with else-branches
- Classic cryptographic primitives (symmetric and asymmetric encryption, signature, pairing, hash)

Include:

- E-passport protocols
- Private authentication protocols

Implementation

Alpha version:

APTE: Algorithm for Proving Trace Equivalence

Content:

- Front-end similar to ProVerif
- Handle equivalence between processes without replication
- Display witness of non-equivalence
- Handle equivalences between sequences of messages
- Handle reachability properties for processes without replication

Outline

1. Proving more equivalence with ProVerif
2. Decision procedure for trace equivalence
3. Composing trace equivalence

Motivation

Concrete example: e-passport protocols

- Basic Access Control (BAC) : establishes session keys between a reader and a passport
- Passive Authentication (PA)
- Active Authentication (AA)

Protocols are verified in isolation

Possible problems:

- Protocols may share some keys
- Protocols may share some cryptographic primitives
- Tools may not be able to prove the security property

Result

Verifying **S** on **P**

and

Verifying **S** on **Q**

↓ under conditions

Verifying **S** on **P** and **Q** running in parallel

where

- **P** and **Q** may share secrets and cryptographic primitives
- **S** is a security property

Conditions of the result

- The shared secret keys are not revealed
- The protocols **P** and **Q** are tagged
- The public keys are revealed at the beginning

Conclusion

- Relations between different notions of equivalences

- Algorithms to prove equivalence

  - Extension of ProVerif for proving more equivalence
  - New decision procedure for trace equivalence (else branches)
  - New automatic tool : APTE

  ✓ Anonymity for the private authentication protocol
  ✓ Unlinkability for the e-passport protocol

- Composition result for trace equivalence

  Application on the e-passport protocol

Future Works

- Improve the algorithms
  - Prove unlinkability of the e-passport protocol with ProVerif for unbounded number of sessions
  - Add cryptographic primitives in APTE (e.g. blind signature, xor, re-encryption)
- Optimize the implementation of APTE
  - Distributed implementation
  - Interleaving problem
  - Simplify the strategy on the rules
- Sequential composition for trace equivalence