Proving more observational equivalences with ProVerif

B.Blanchet & V.Cheval

26 July 2012

CONTEXT

- Cryptographic protocols

Most communications take place over a public network

Cryptographic protocols

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

It important to verify their security

CONTEXT

- Reliable cryptography
- Correct specification
- Implementation satisfying the specification

CONTEXT

- Reliable cryptography
- Correct specification
- Implementation satisfying the specification

- Some security properties

CONTEXT

- Reliable cryptography
- Correct specification
- Implementation satisfying the specification

- Some security properties

Reachability properties

- Secrecy, Authentication, ...

CONTEXT

- Reliable cryptography
- Correct specification
- Implementation satisfying the specification

- Some security properties

Reachability properties

- Secrecy, Authentication, ...

Equivalence properties

- Anonymity, Privacy, Receipt-Freeness, ...

CONTEXT

- Formalism

Alice

Intruder

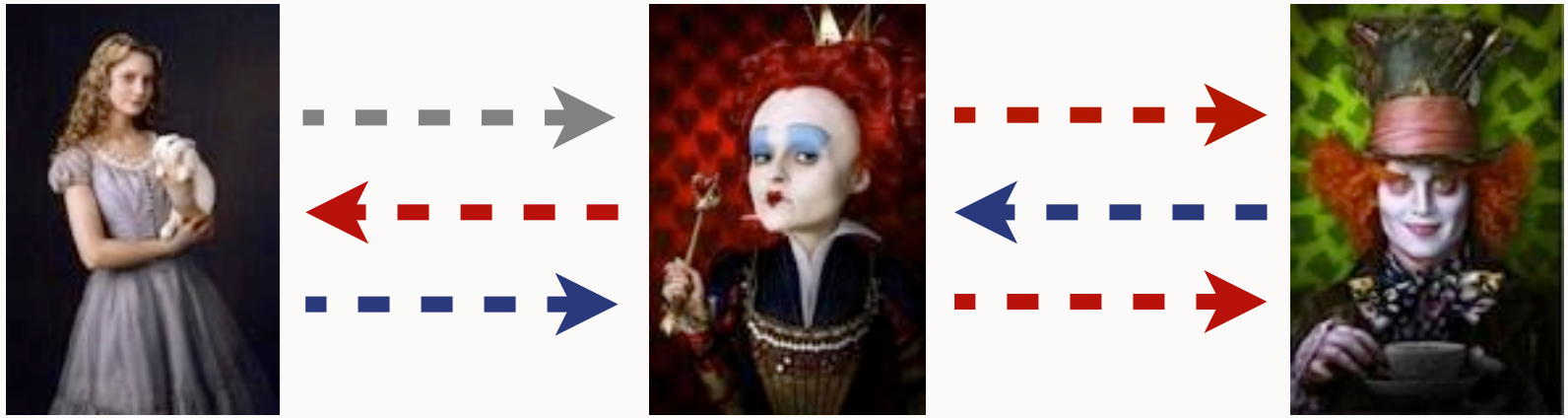
Bob

The intruder can

- intercept all messages
- transmit or modify messages
- test equality between messages
- initiate several sessions

CONTEXT

- Formalism

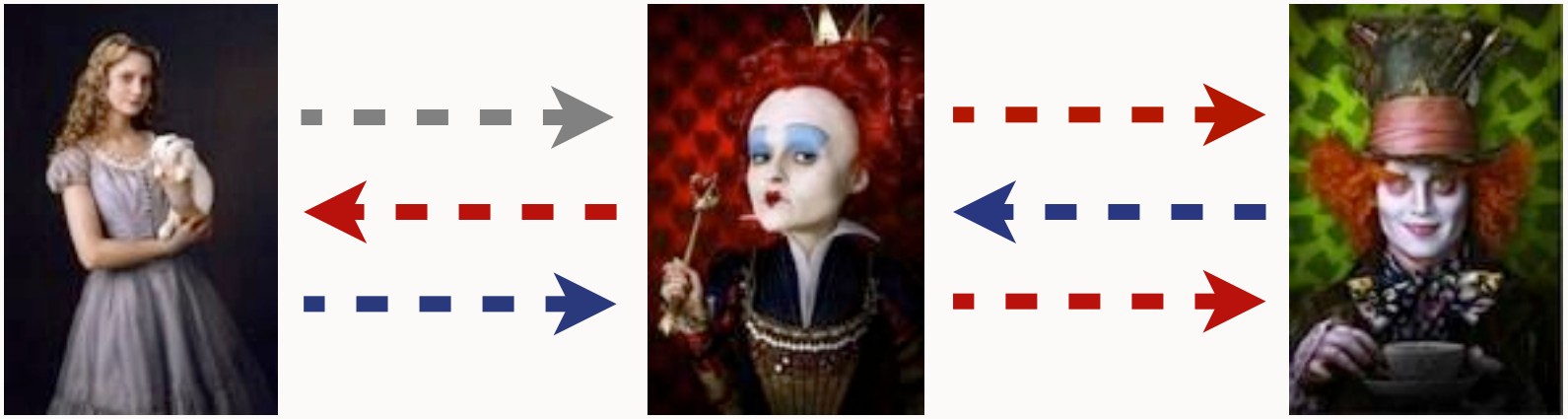


The intruder can

- intercept all messages
- transmit or modify messages
- test equality between messages
- initiate several sessions

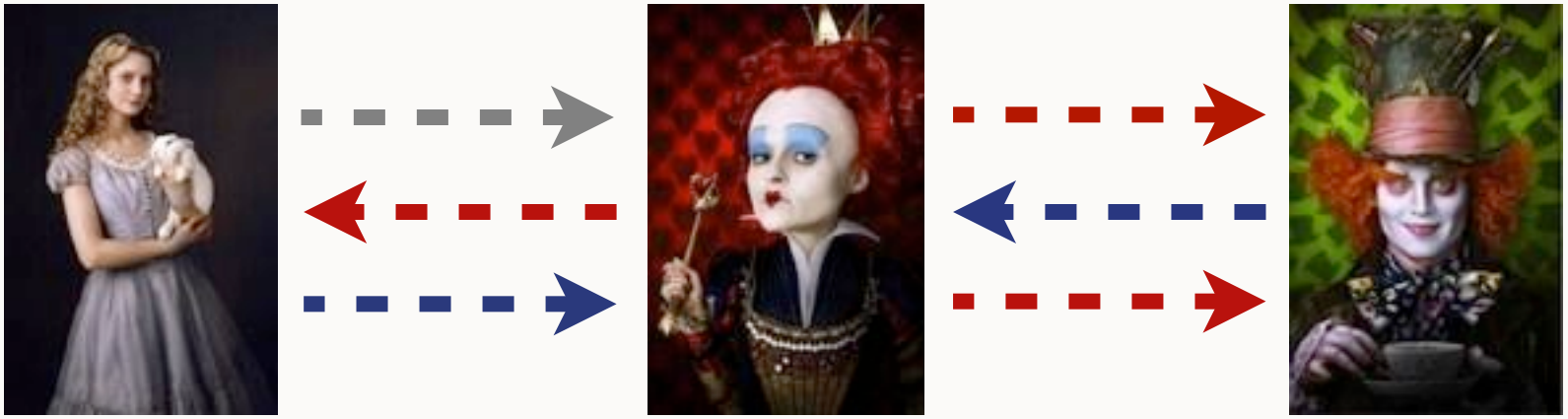
CONTEXT

- Reachability properties : secrecy, authentication,...



CONTEXT

- Reachability properties : secrecy, authentication,...

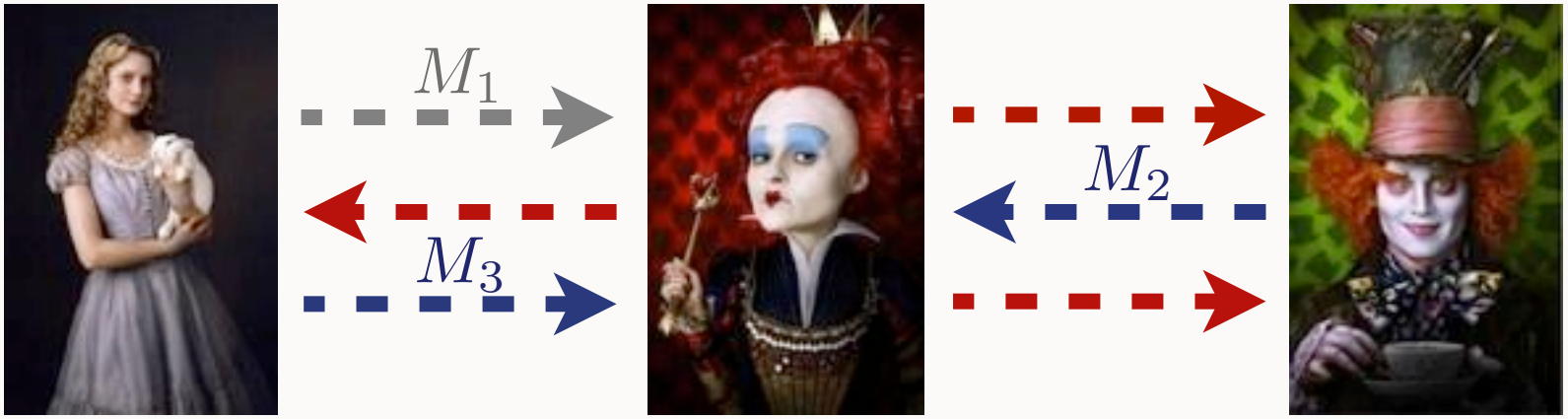


Can the intruder deduce Alice's secret ?

CONTEXT

- Reachability properties : secrecy, authentication,...

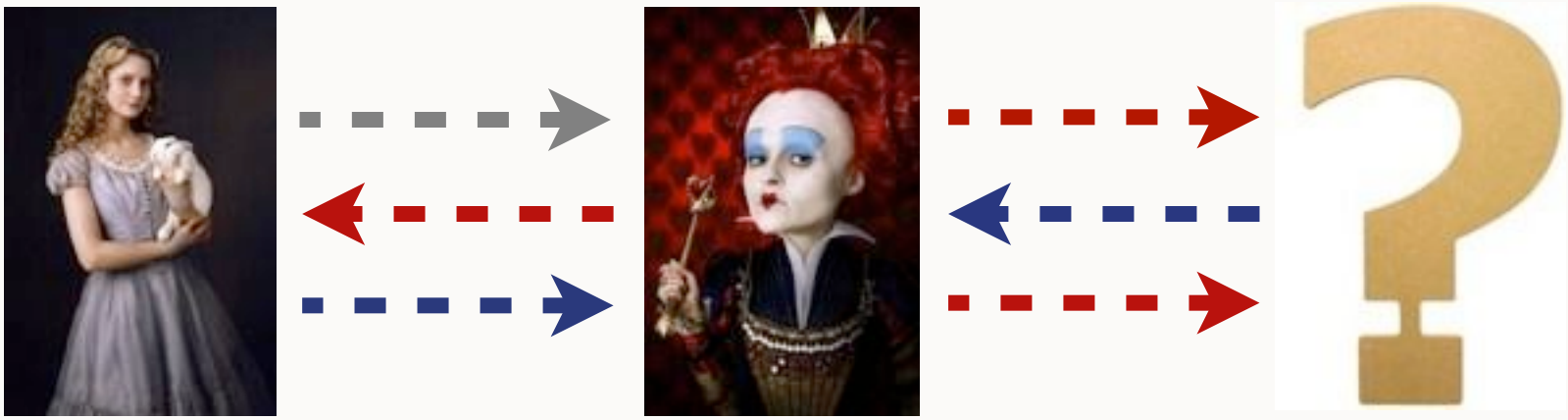
intruder's knowledge : M_1 M_2 M_3 + basic knowledge



Can the intruder deduce Alice's secret ?

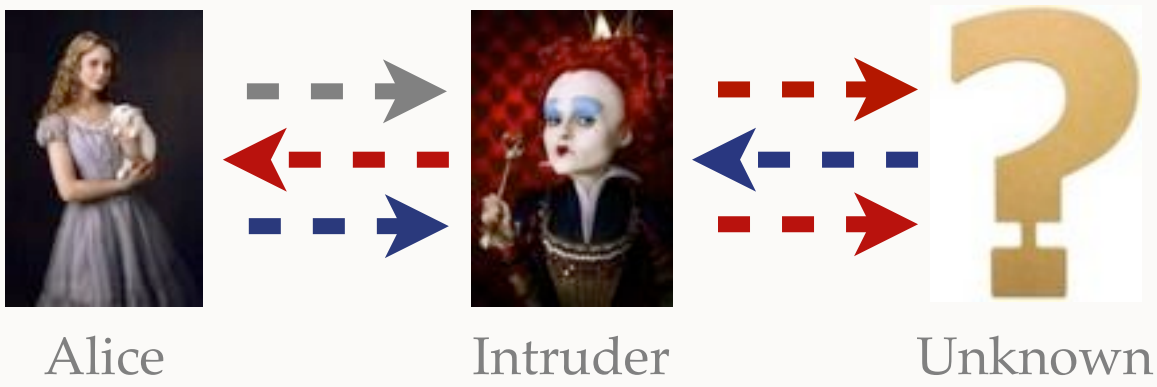
CONTEXT

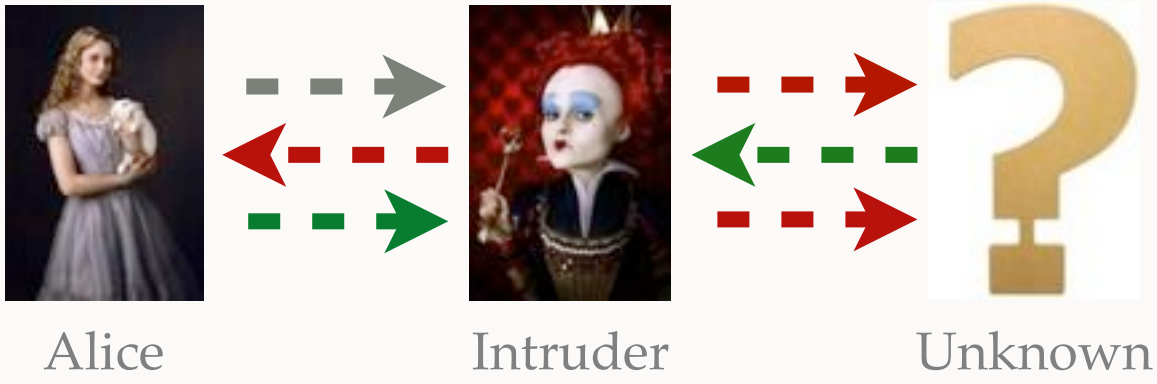
- Equivalence properties : strong secret, anonymity,...



CONTEXT

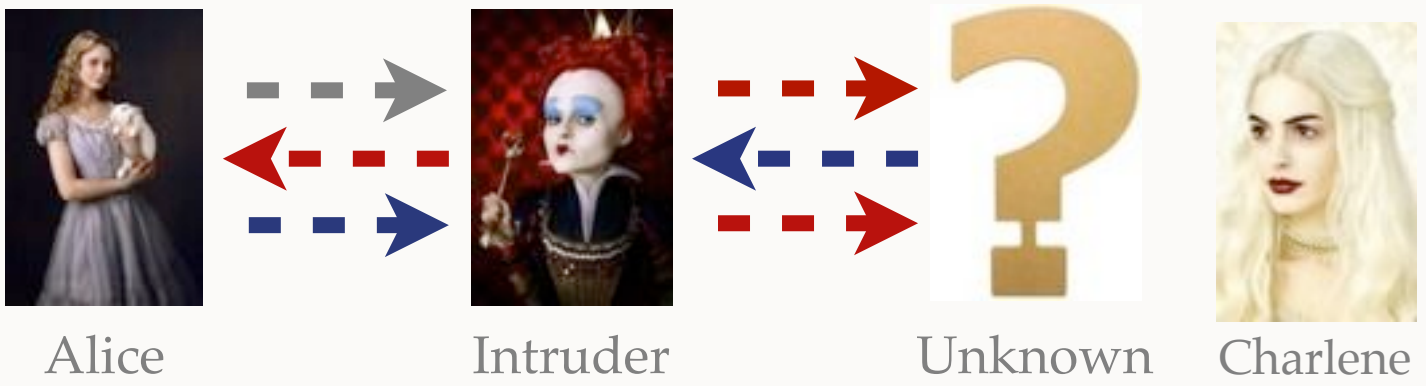
- Equivalence properties : strong secret, anonymity,...

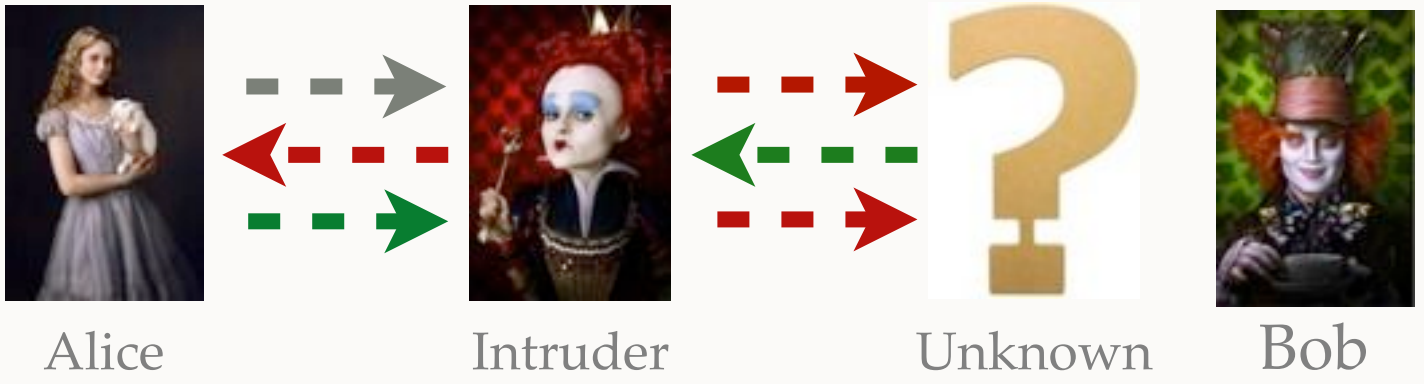




CONTEXT

- Equivalence properties : strong secret, anonymity,...





PREVIOUS WORKS

Observational Equivalence

- A. Tiu and J. E. Dawson. *Automating open bisimulation checking for the spi calculus.*
- M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires.* Phd thesis
- B. Blanchet, M. Abadi, and C. Fournet. *Automated verification of selected equivalences for security protocols.*

➡ Tool : B. Blanchet, *ProVerif*

Trace Equivalence

- V. Cheval, H. Comon-Lundh, and S. Delaune. *Trace equivalence decision: Negative tests and non-determinism.*
- S. Ciobâca. *Phd Thesis*

MOTIVATION

- Example : Private authentication protocol

MOTIVATION

- Example : Private authentication protocol

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

Alice

Bob

MOTIVATION

- Example : Private authentication protocol

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

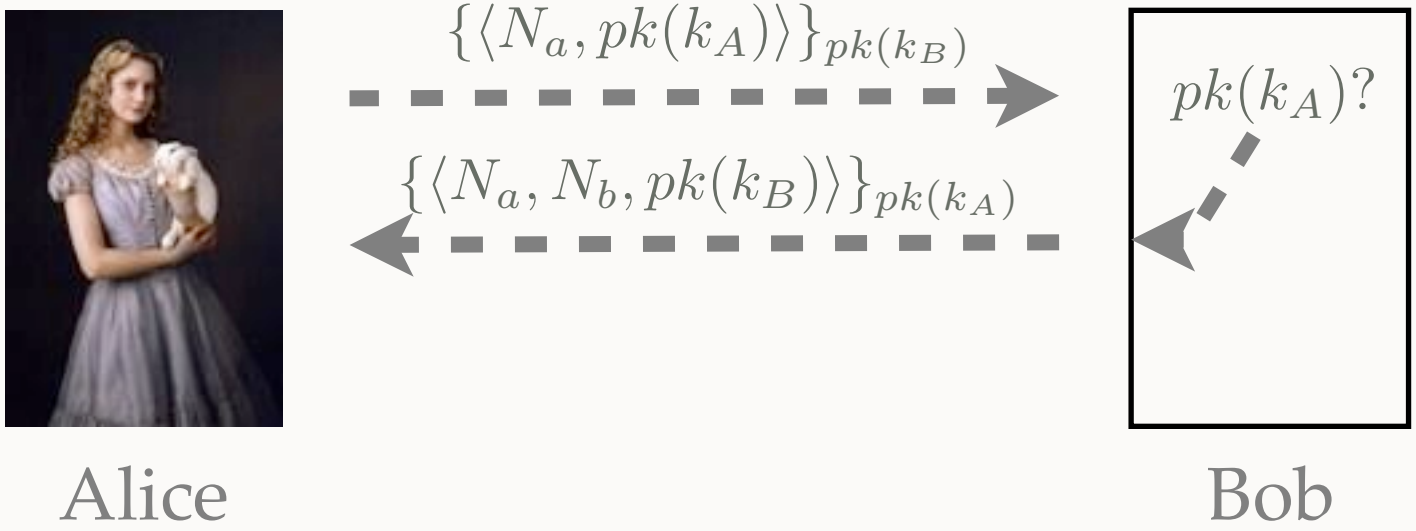
$pk(k_A)?$

Alice

Bob

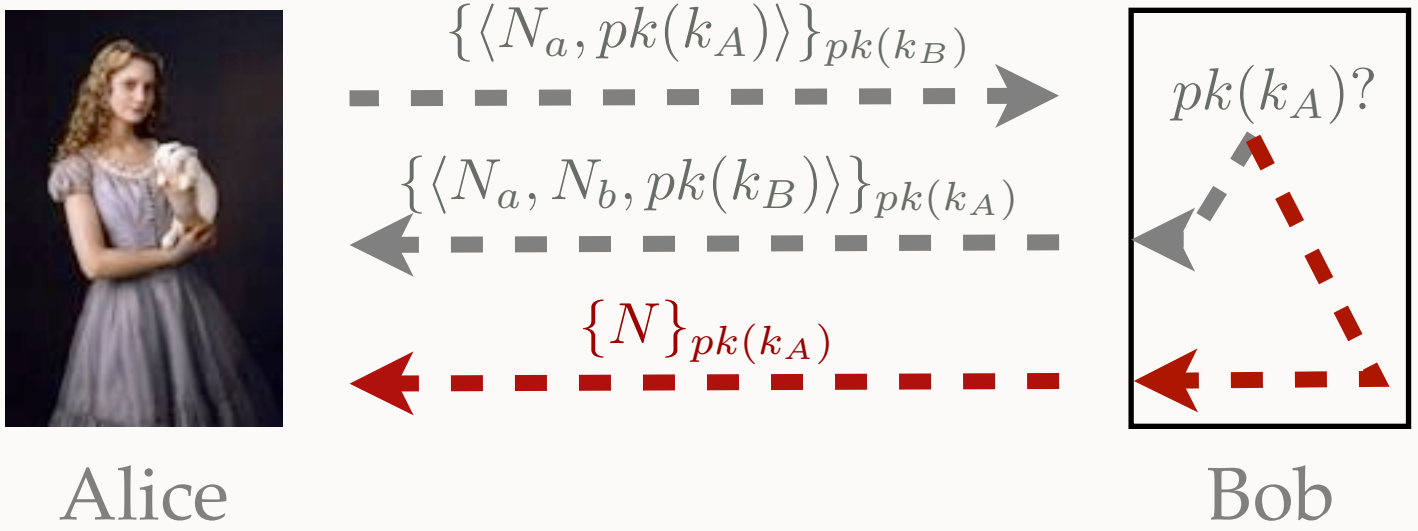
MOTIVATION

- Example : Private authentication protocol



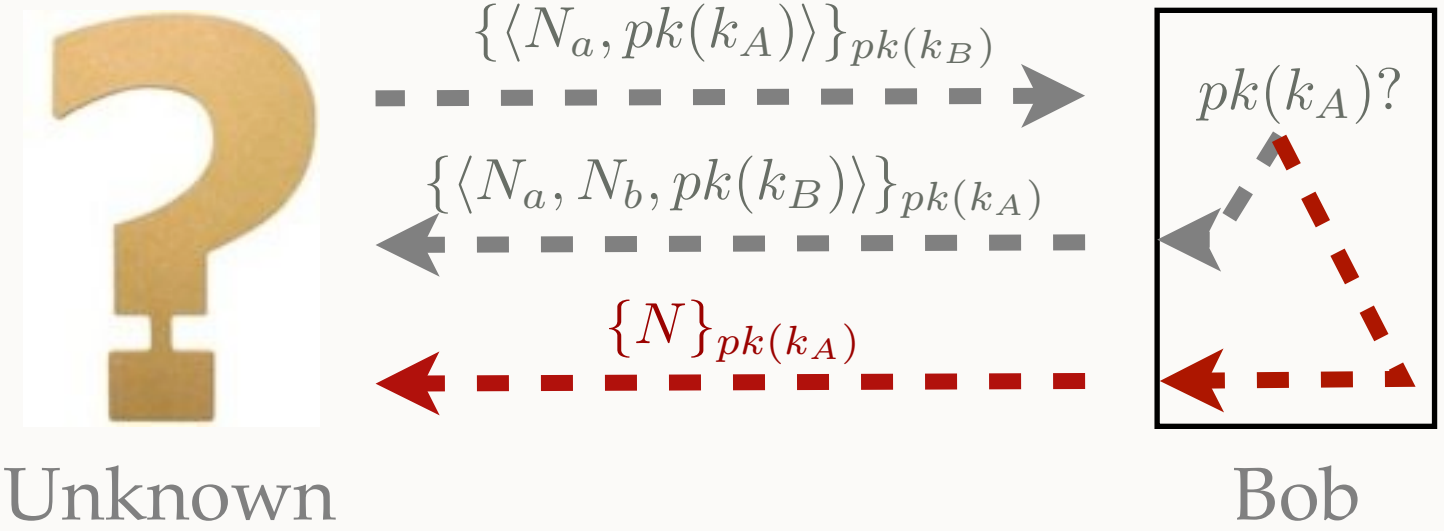
MOTIVATION

- Example : Private authentication protocol



MOTIVATION

- Example : Private authentication protocol



MOTIVATION

- Example

Alice

Intruder

Bob

Charlene

Intruder

Bob

MOTIVATION

- Example

Alice

Bob

Charlene

Bob

MOTIVATION

- Example

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

Alice

Bob

Charlene

Bob

MOTIVATION

- Example

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

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

Alice

Bob

Charlene

Bob

MOTIVATION

- Example

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

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

$pk(k_A) = y$

Alice

Bob

Charlene

Bob

MOTIVATION

- Example

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

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

$pk(k_A) = y$

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

Alice

Bob

Charlene

Bob

MOTIVATION

- Example

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

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

$pk(k_A) = y$

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

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

Alice

Bob

Charlene

Bob

MOTIVATION

- Example

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

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

$pk(k_A) = y$

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

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

Alice

Bob

$\{\langle N_c, pk(k_C)\rangle\}_{pk(k_B)}$

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

$pk(k_C) = y$

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

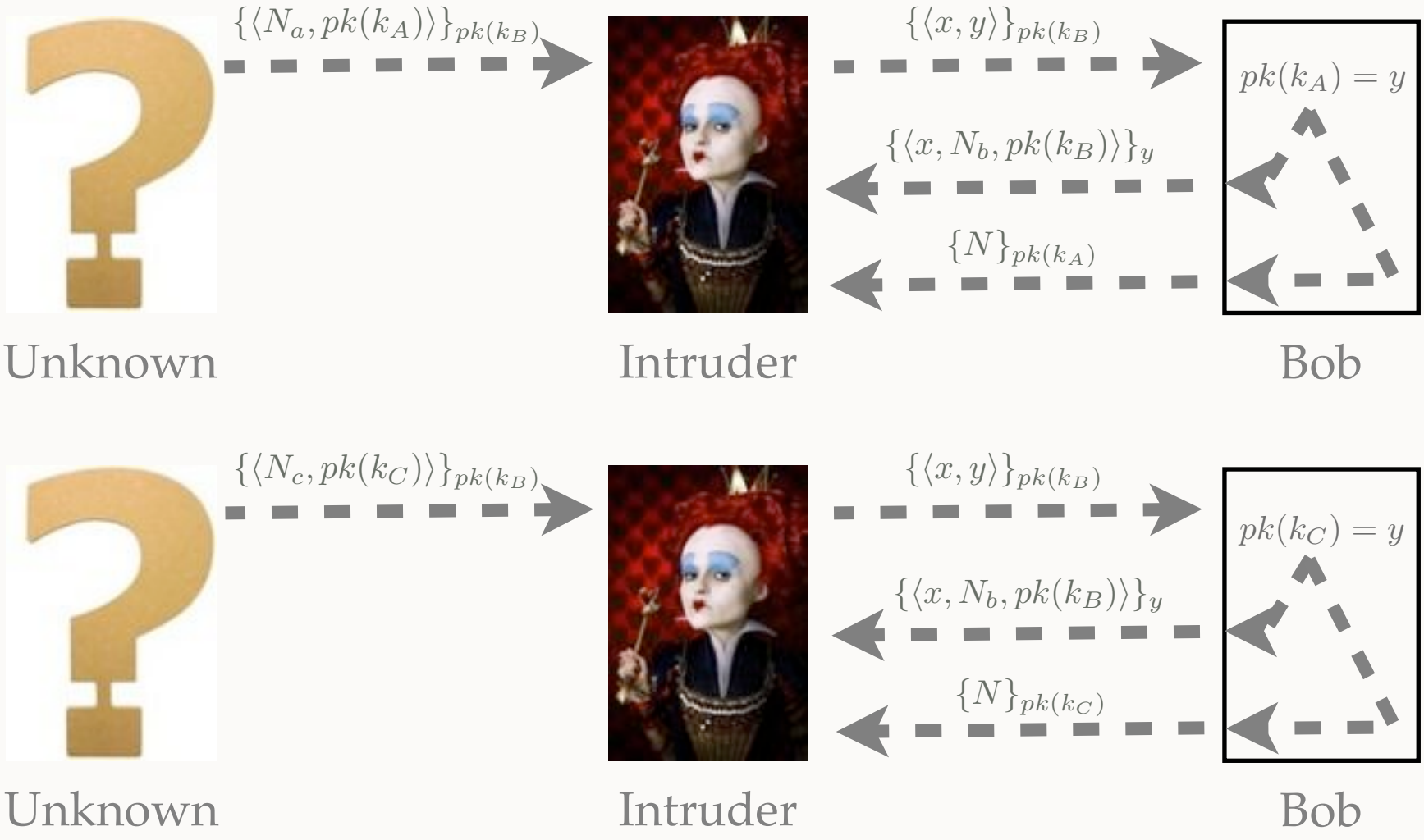
$\{N\}_{pk(k_C)}$

Charlene

Bob

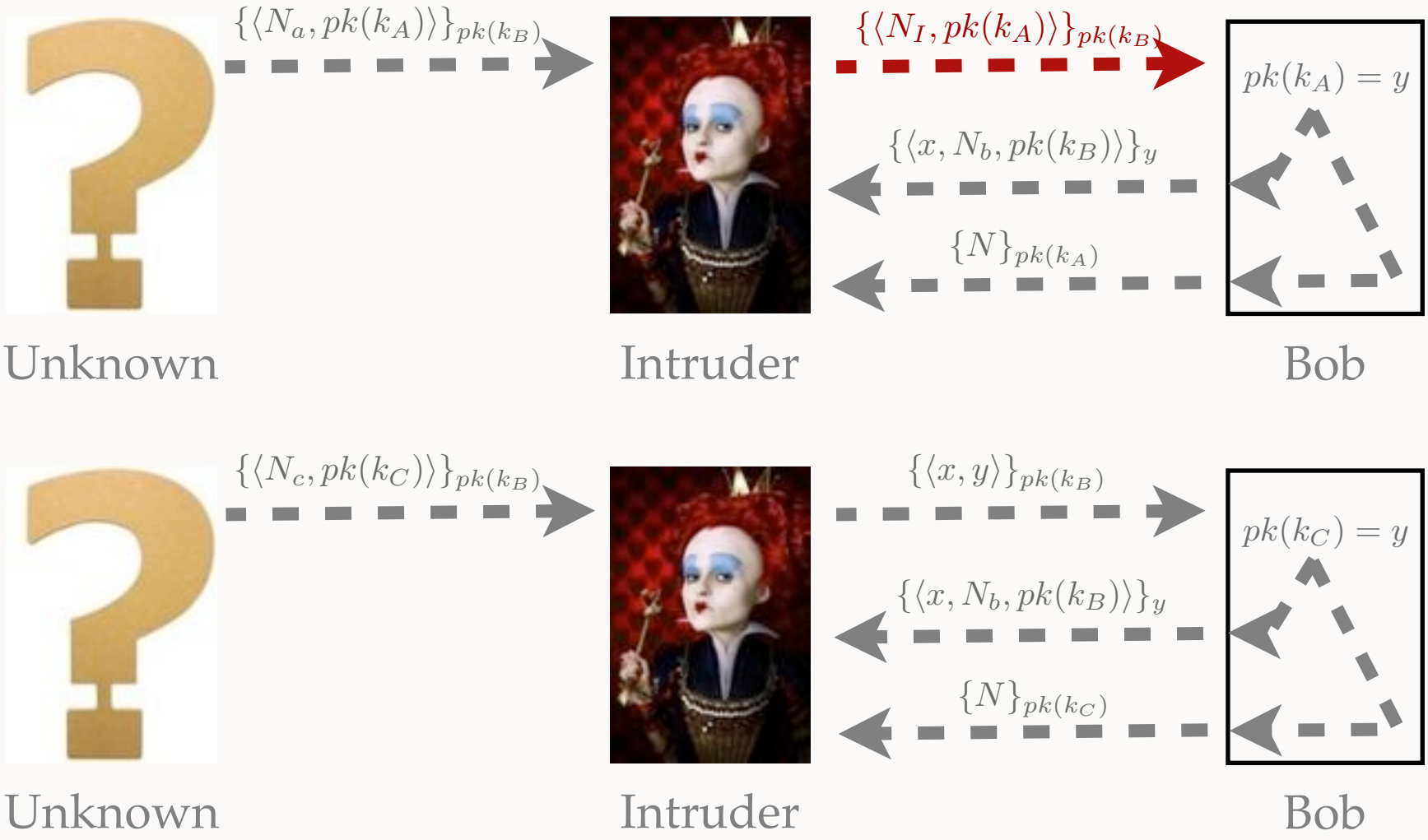
MOTIVATION

- Example



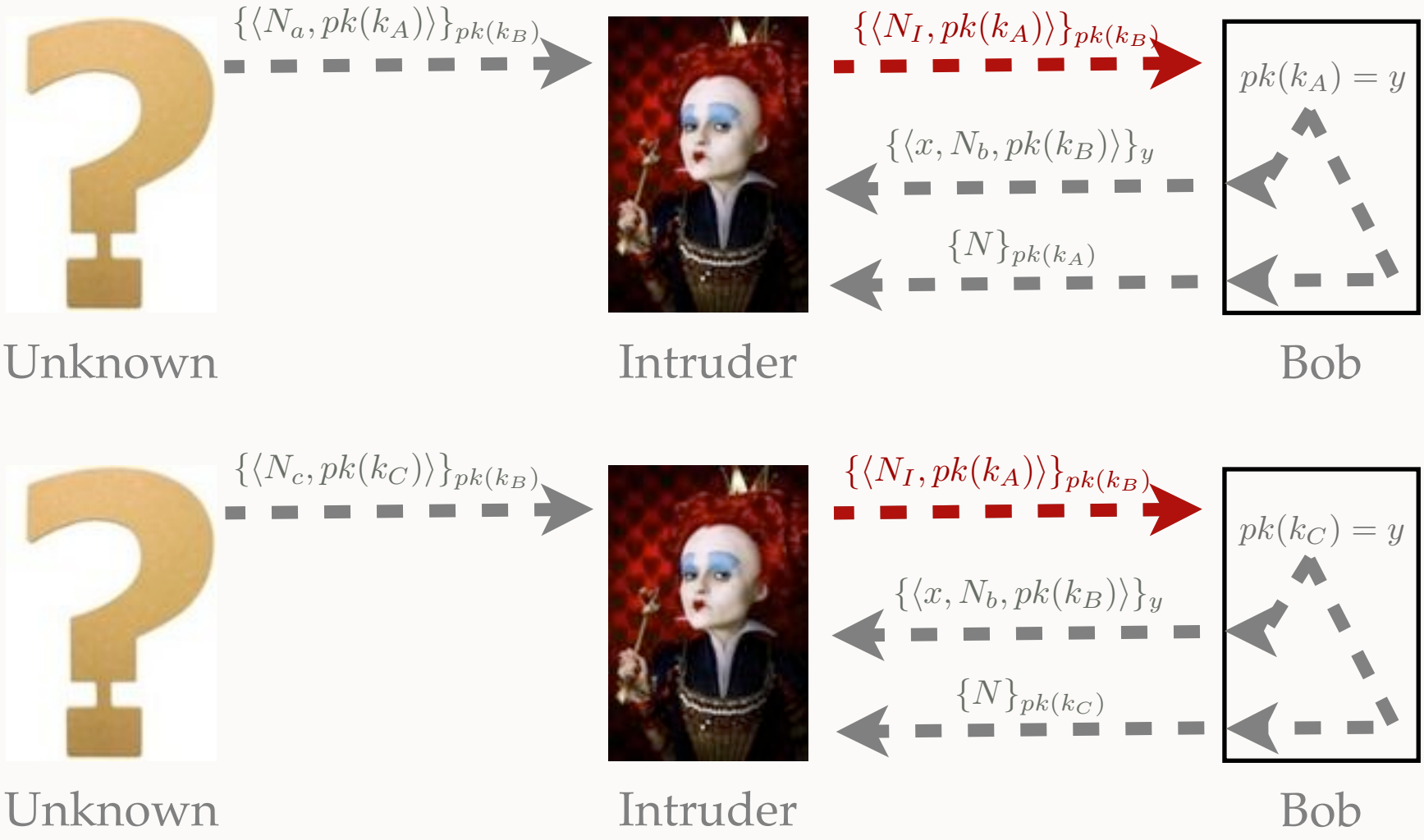
MOTIVATION

- Example



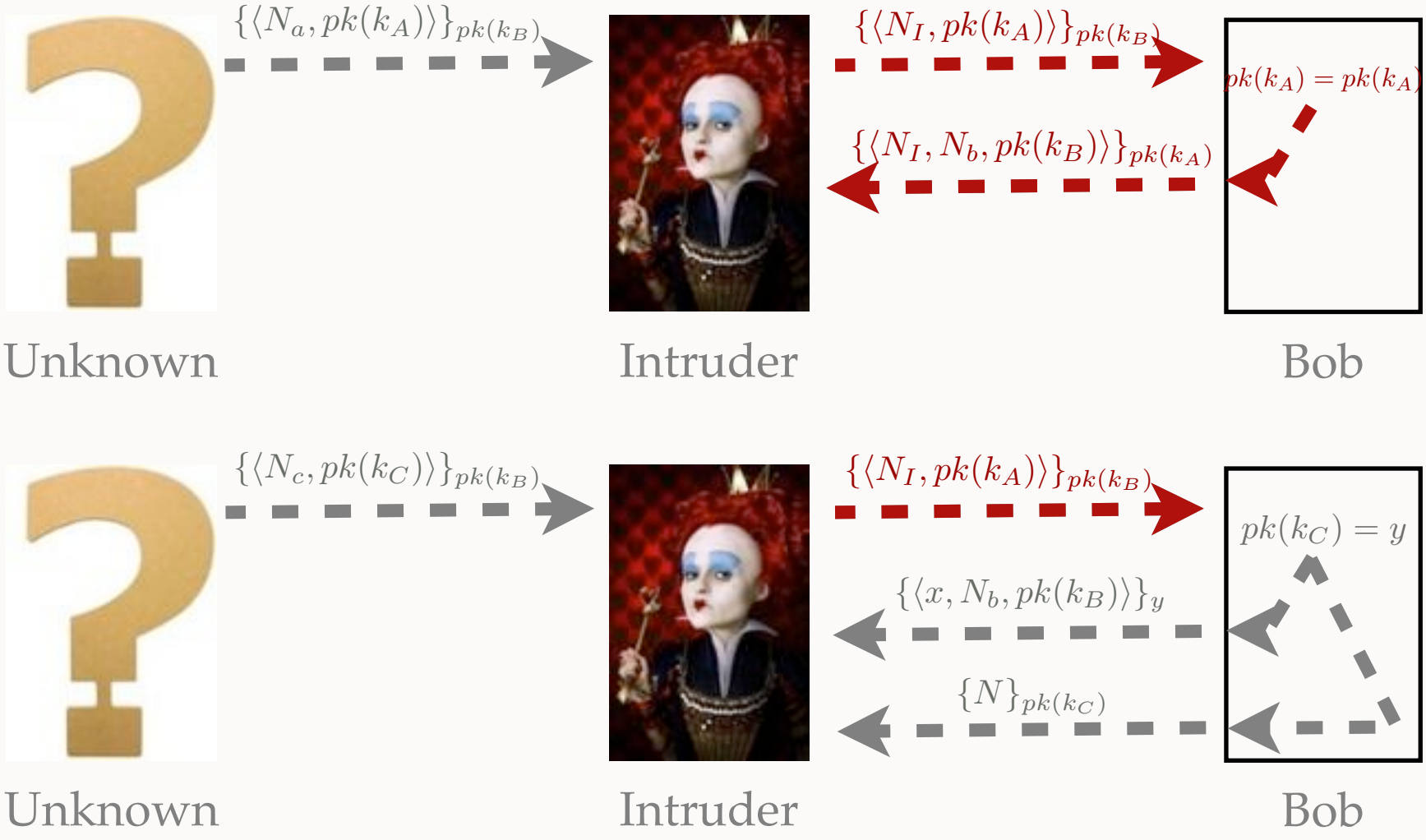
MOTIVATION

- Example



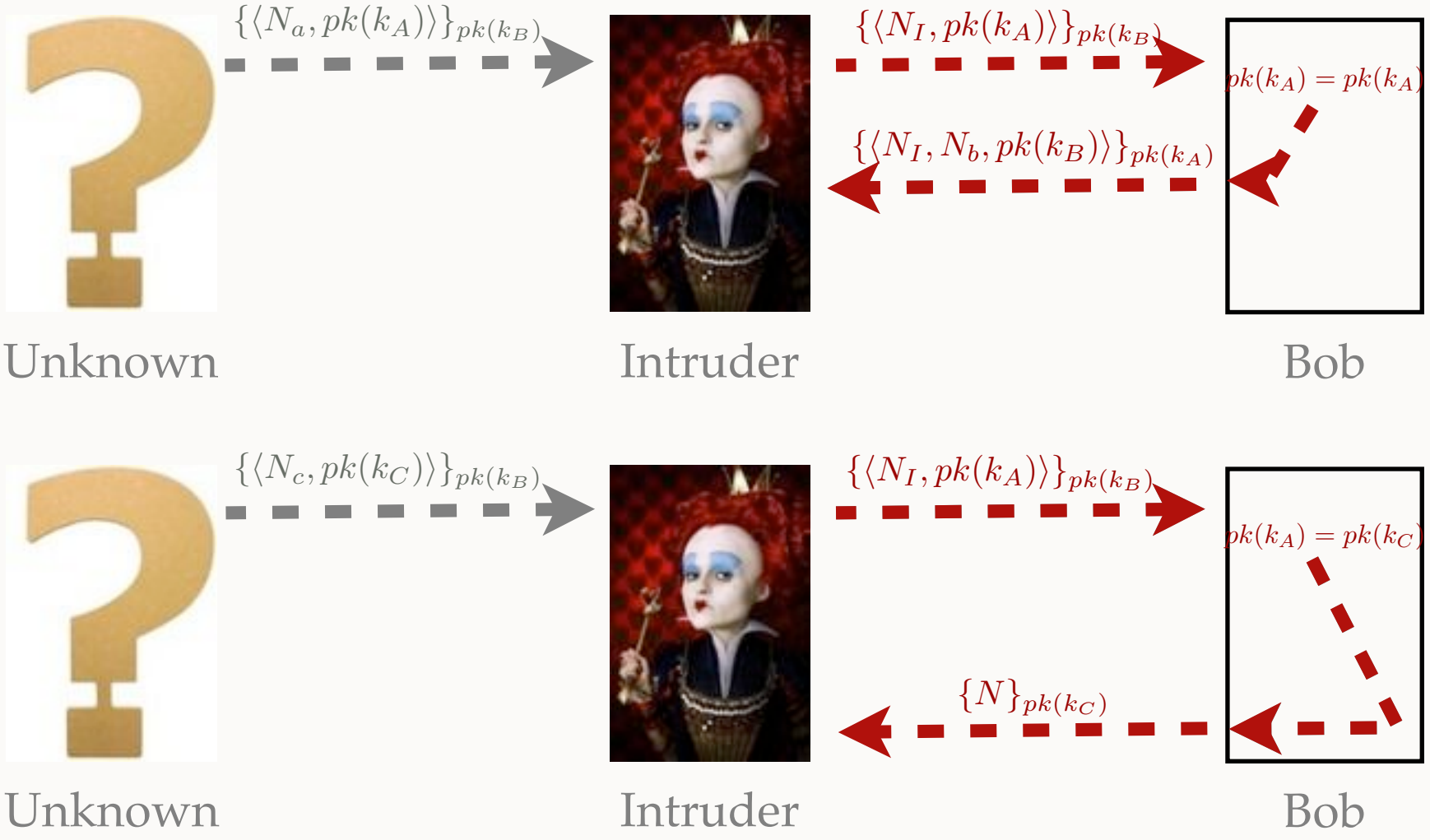
MOTIVATION

- Example



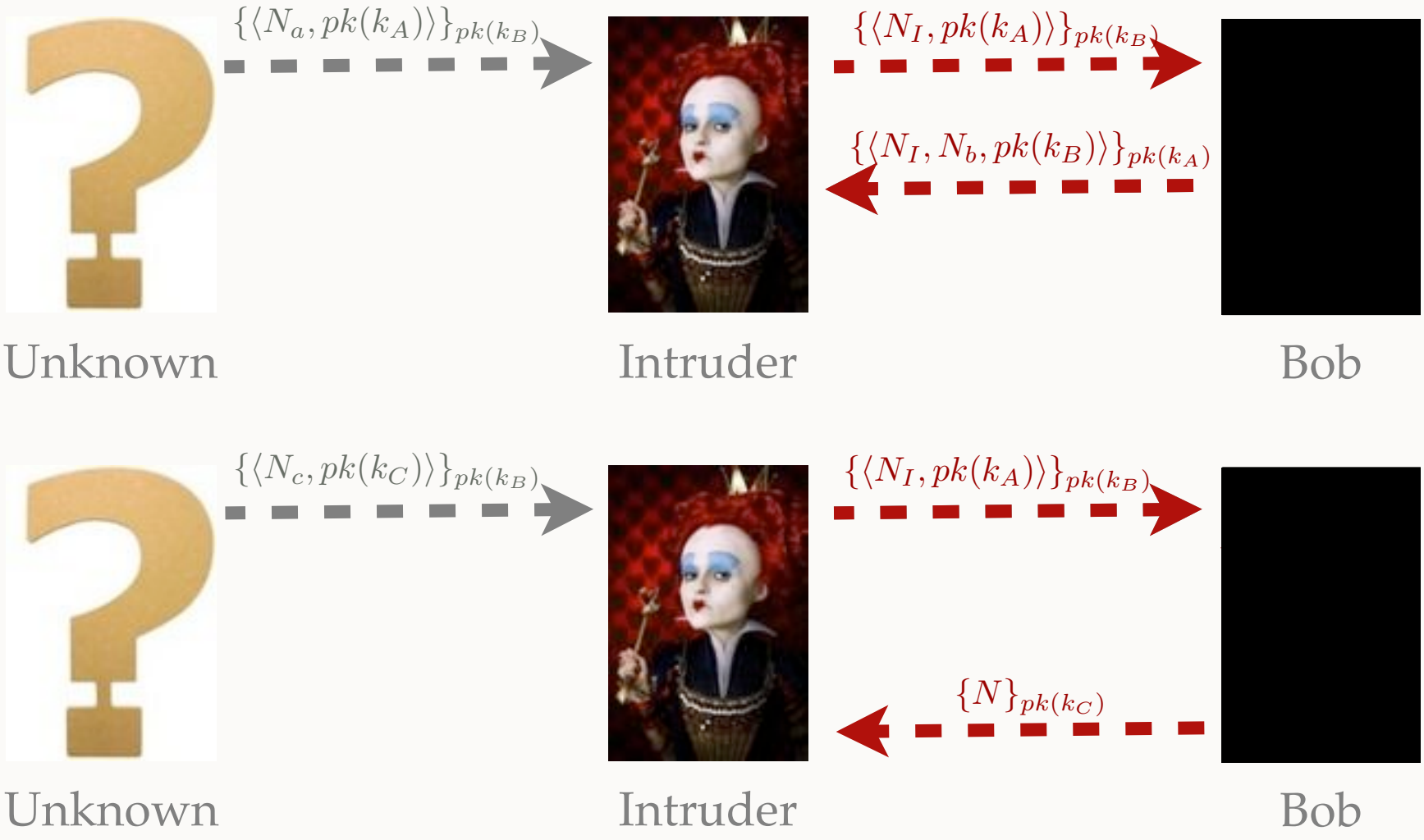
MOTIVATION

- Example



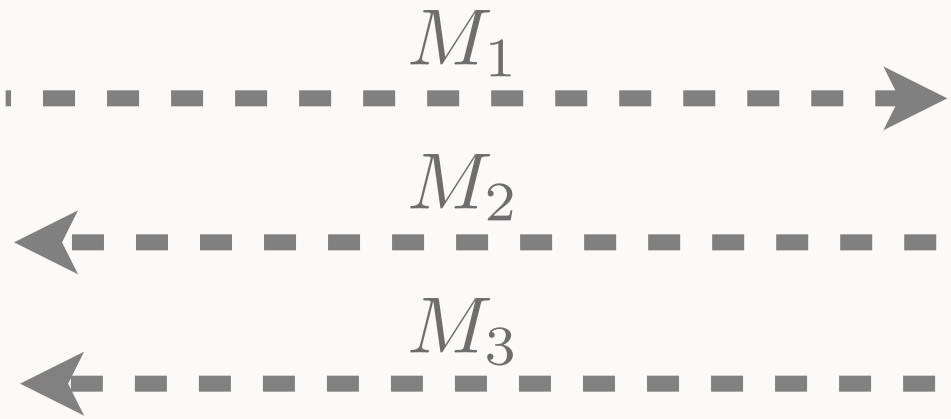
MOTIVATION

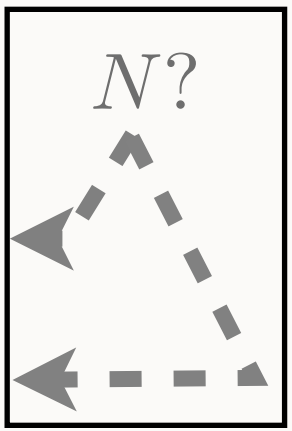
- Example



PROVERIF

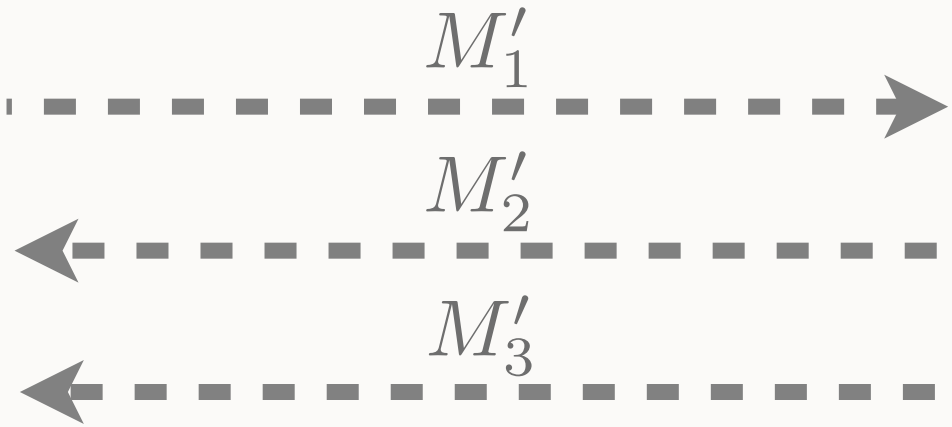
- Biprocesses

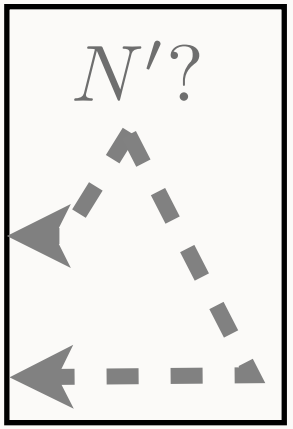




Alice

Bob



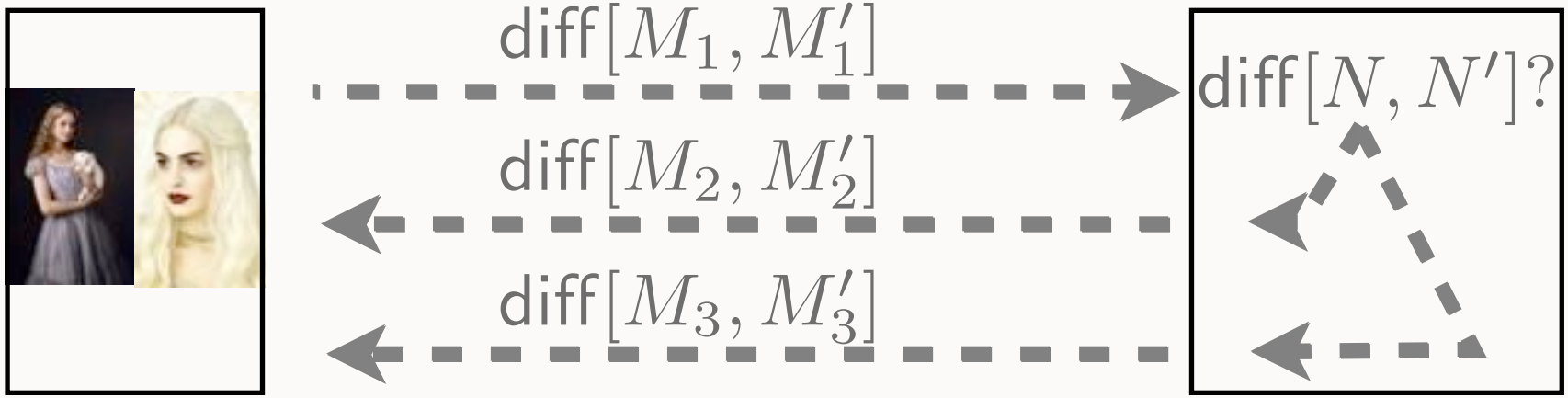


Charlene

Bob

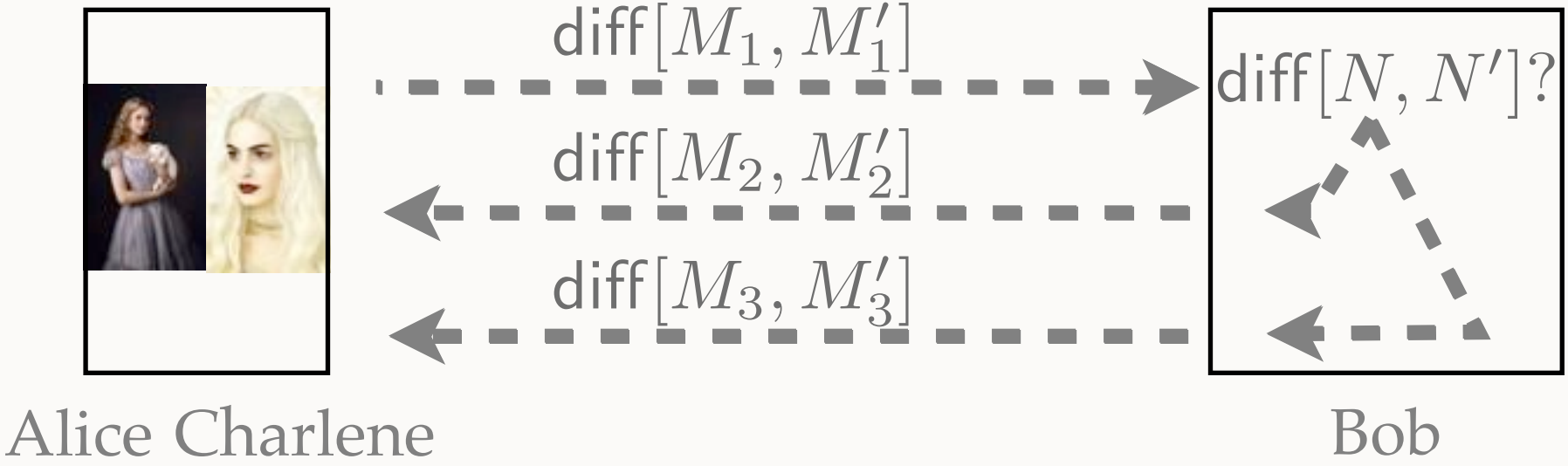
PROVERIF

- Biprocesses



PROVERIF

- Biprocesses



Biprocesses force matching of some traces

PROVERIF

- Was first an analyzer for reachability properties
- Based on Horn clauses
- Equational theory : constructors

$$\mathsf{dec}(\{x\}_y, y) =_E x$$

$$\{\mathsf{dec}(x, y)\}_y =_E x$$

- Rewrite rules : destructors

$$\mathsf{dec}(\{x\}_y, y) \to x$$
$$\mathsf{equal}(x, x) \to x$$

CONTRIBUTION

We introduce destructor with tests between terms

$$\mathrm{g}(M_1, \ldots, M_n) \to M \text{ with } \Phi$$

CONTRIBUTION

We introduce destructor with tests between terms

$$\mathrm{g}(M_1, \ldots, M_n) \to M \text{ with } \Phi$$

- The *If-then-else* destructor

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

CONTRIBUTION

We introduce destructor with tests between terms

$$\mathrm{g}(M_1, \ldots, M_n) \to M \text{ with } \Phi$$

- The *If-then-else* destructor

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

- The *dec-else* destructor

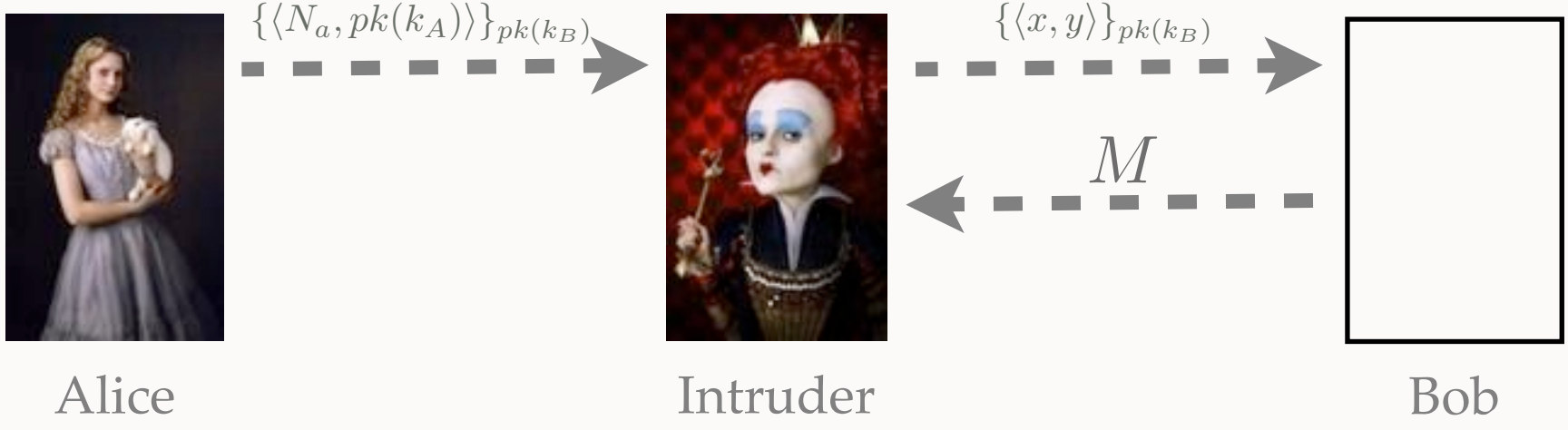
$$\mathsf{dec}(\{x\}_y, y, t) \to x$$
$$\mathsf{dec}(x, y, t) \to t \qquad \text{with } \forall z.\ x \neq \{z\}_y$$

CONTRIBUTION

- Protocol transformation

CONTRIBUTION

- Protocol transformation



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

PROVERIF

What are the modifications in ProVerif algorithm ?

CURRENT PROVERIF

- Horn clauses

CURRENT PROVERIF

- Horn clauses

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

$pk(k_A)?$

Alice

Bob

CURRENT PROVERIF

- Horn clauses

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

$pk(k_A)?$

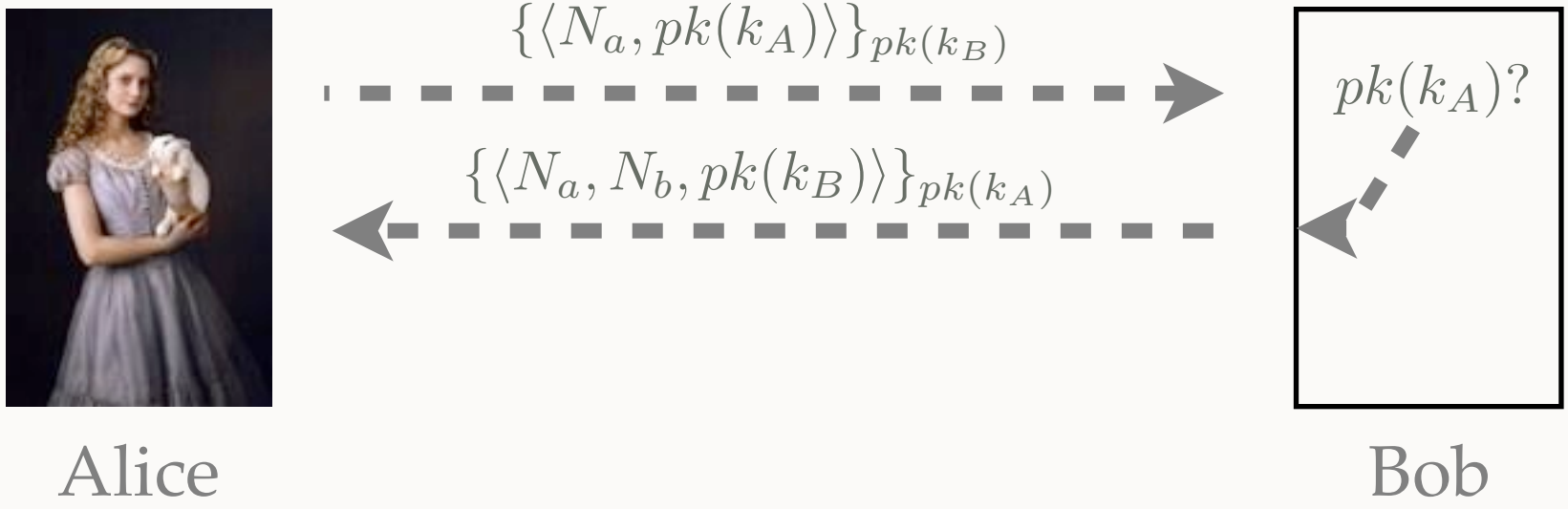
Alice

Bob

$$\rightarrow \mathrm{att}(\{N_a, pk(k_A)\}_{pk(k_B)})$$

CURRENT PROVERIF

- Horn clauses



$$\rightarrow \mathrm{att}(\{N_a, pk(k_A)\}_{pk(k_B)})$$

CURRENT PROVERIF

- Horn clauses

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

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

$pk(k_A)?$

Alice

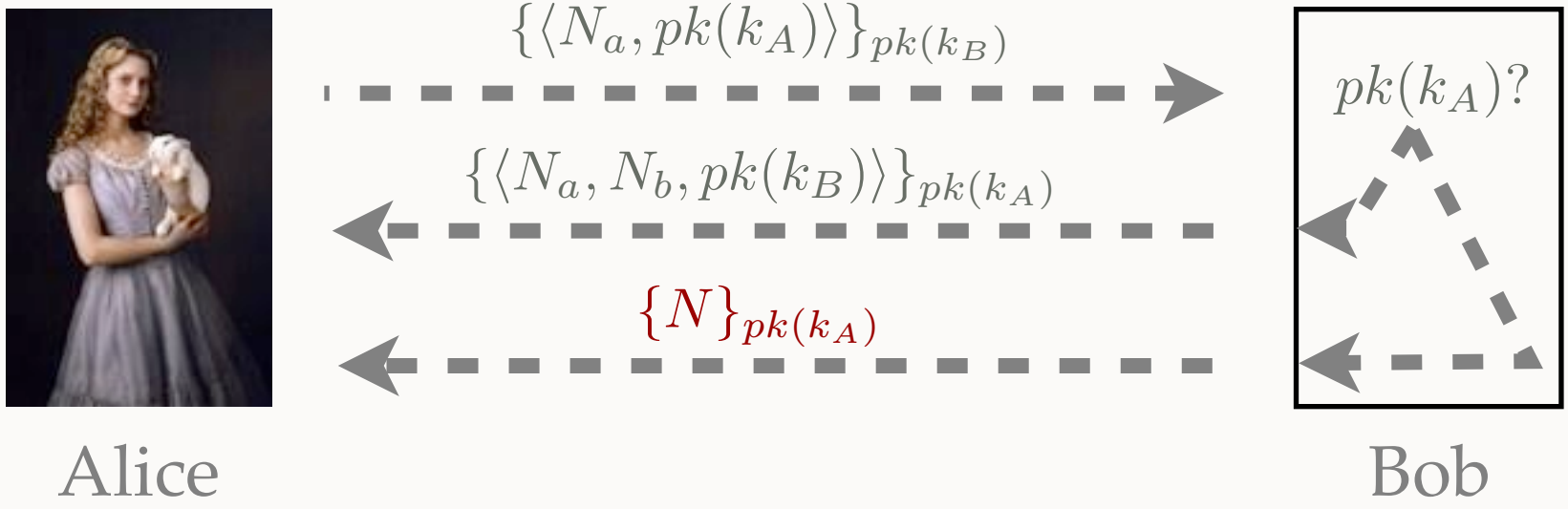
Bob

$$\rightarrow \mathrm{att}(\{N_a, pk(k_A)\}_{pk(k_B)})$$

$$\mathrm{att}(\{x, pk(k_A)\}_{pk(k_B)}) \rightarrow \mathrm{att}(\{x, N_b, pk(k_B)\}_{pk(k_A)})$$

CURRENT PROVERIF

- Horn clauses

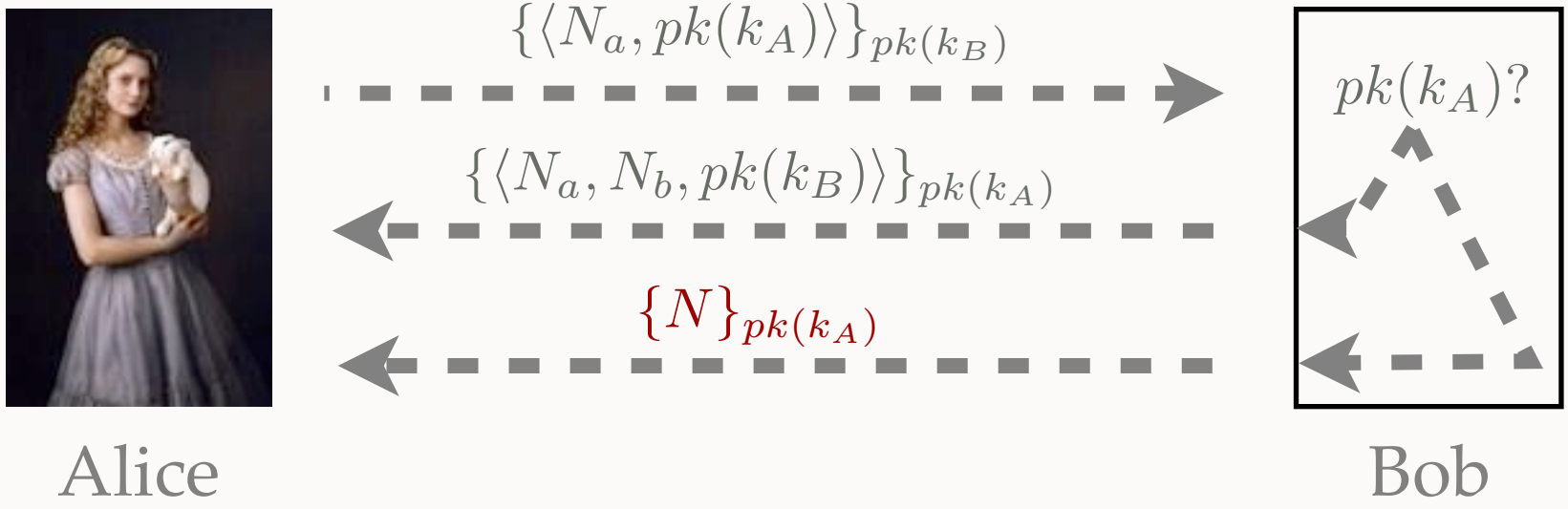


$$\rightarrow \mathrm{att}(\{N_a, pk(k_A)\}_{pk(k_B)})$$

$$\mathrm{att}(\{x, pk(k_A)\}_{pk(k_B)}) \rightarrow \mathrm{att}(\{x, N_b, pk(k_B)\}_{pk(k_A)})$$

CURRENT PROVERIF

- Horn clauses



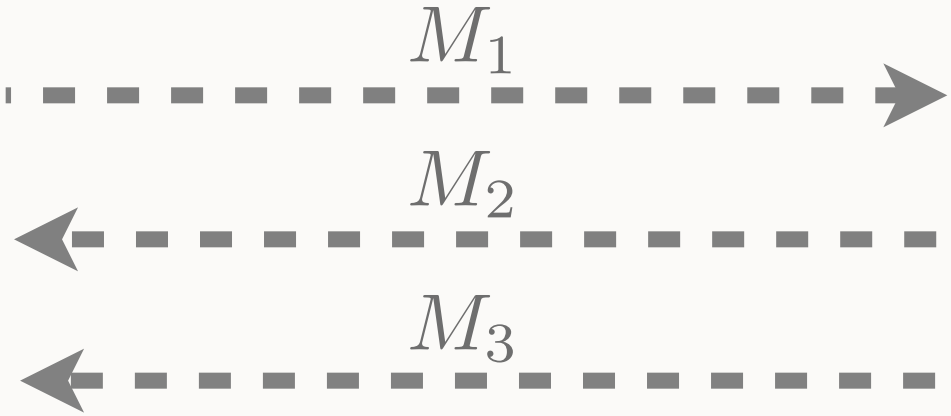
$$\rightarrow \mathrm{att}(\{N_a, pk(k_A)\}_{pk(k_B)})$$

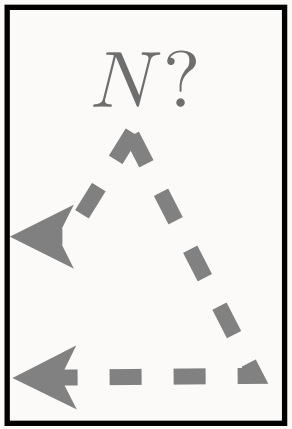
$$\mathrm{att}(\{x, pk(k_A)\}_{pk(k_B)}) \rightarrow \mathrm{att}(\{x, N_b, pk(k_B)\}_{pk(k_A)})$$

$$\mathrm{att}(x) \wedge \forall y.x \neq \{y, pk(k_A)\}_{pk(k_B)} \rightarrow \mathrm{att}(\{N\}_{pk(k_A)})$$

CURRENT PROVERIF

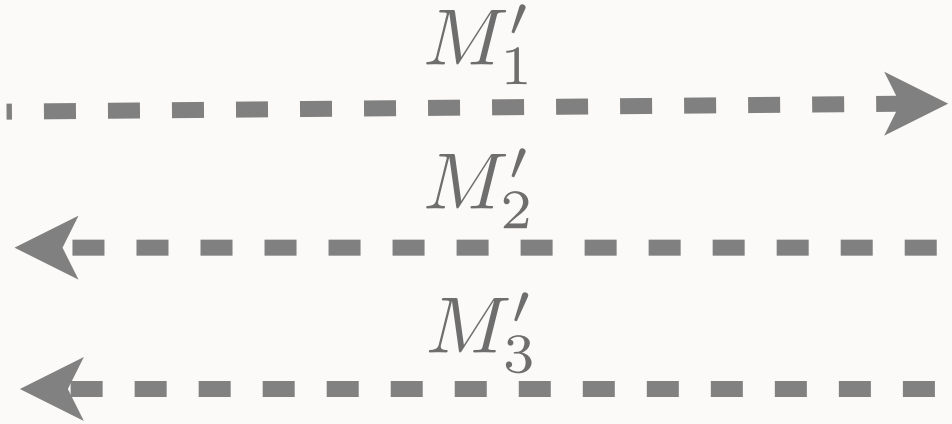
- Biprocess

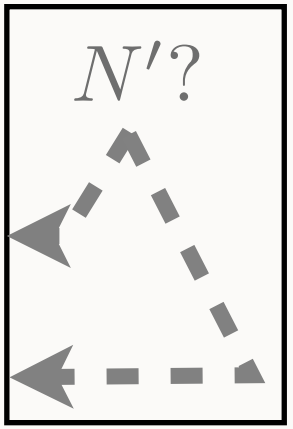




Alice

Bob



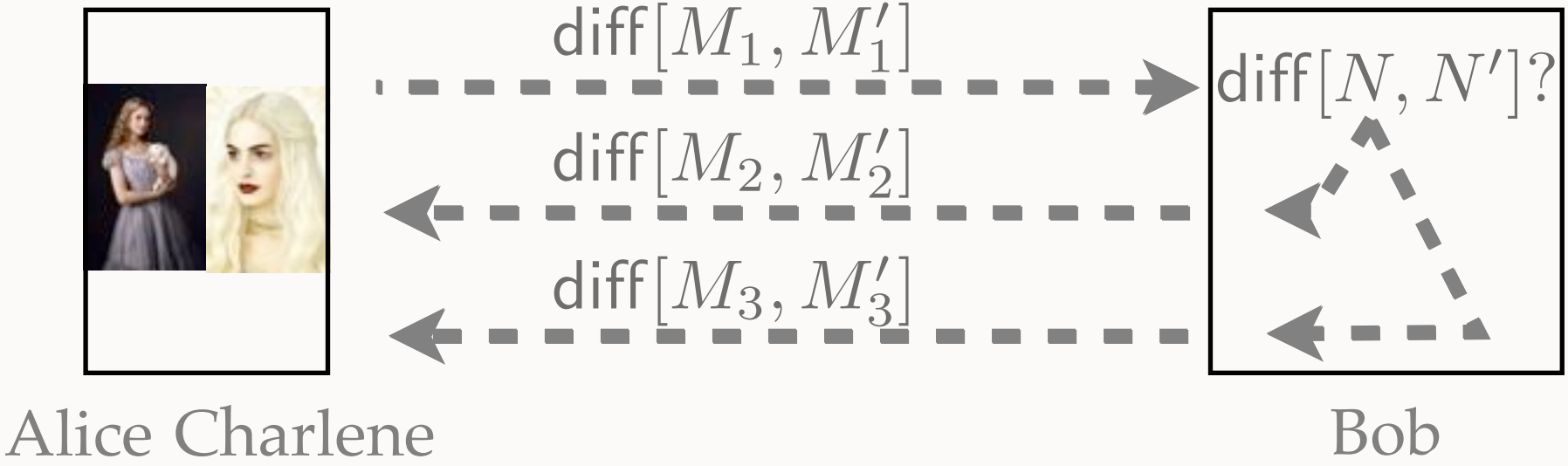


Charlene

Bob

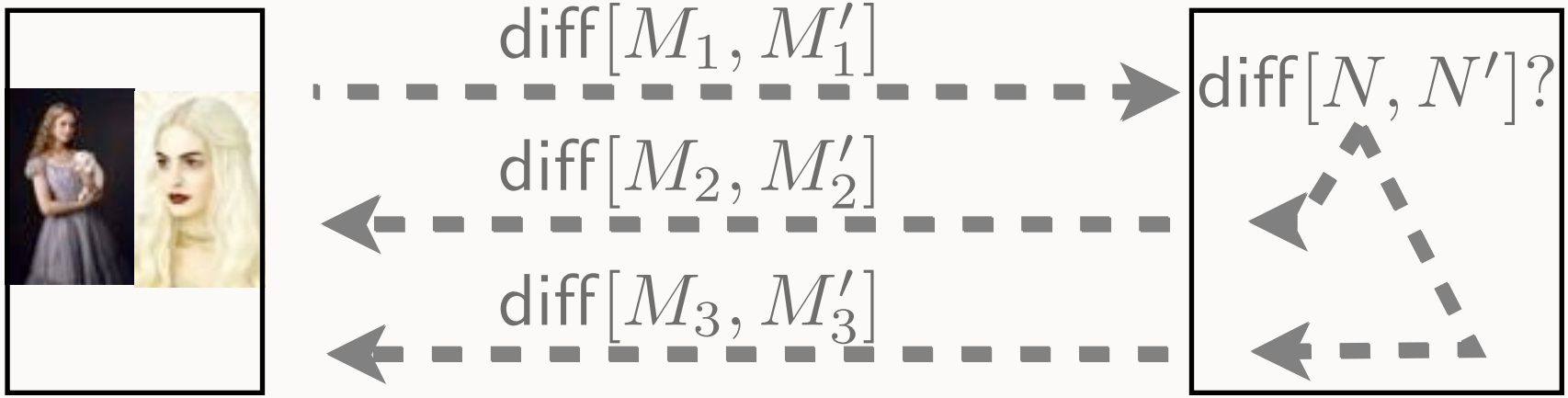
CURRENT PROVERIF

- Biprocess



CURRENT PROVERIF

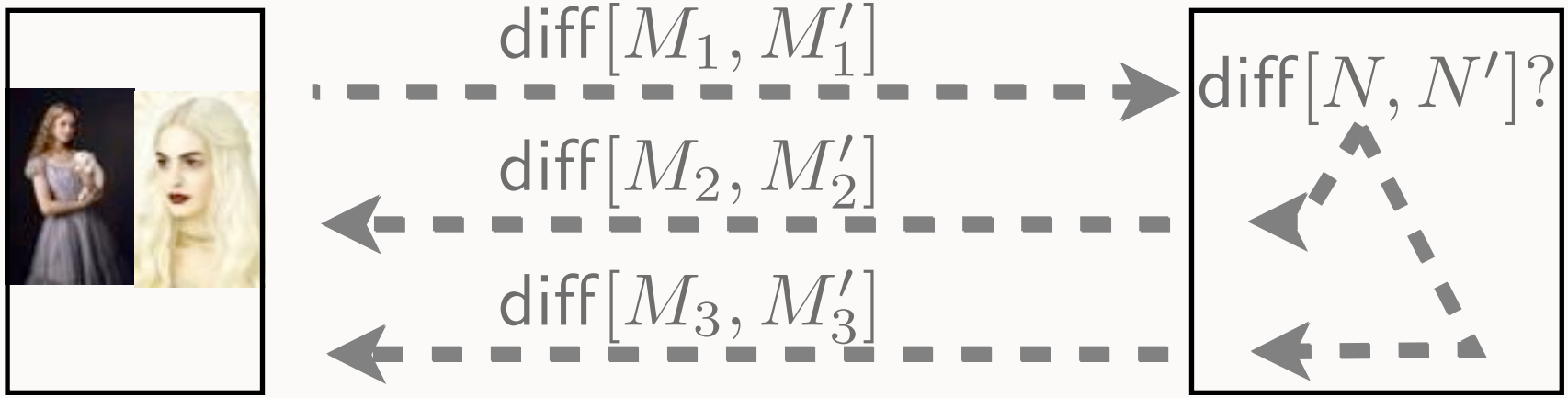
- Biprocess



$\rightarrow \mathrm{att}(M_1, M_1')$

CURRENT PROVERIF

- Biprocess

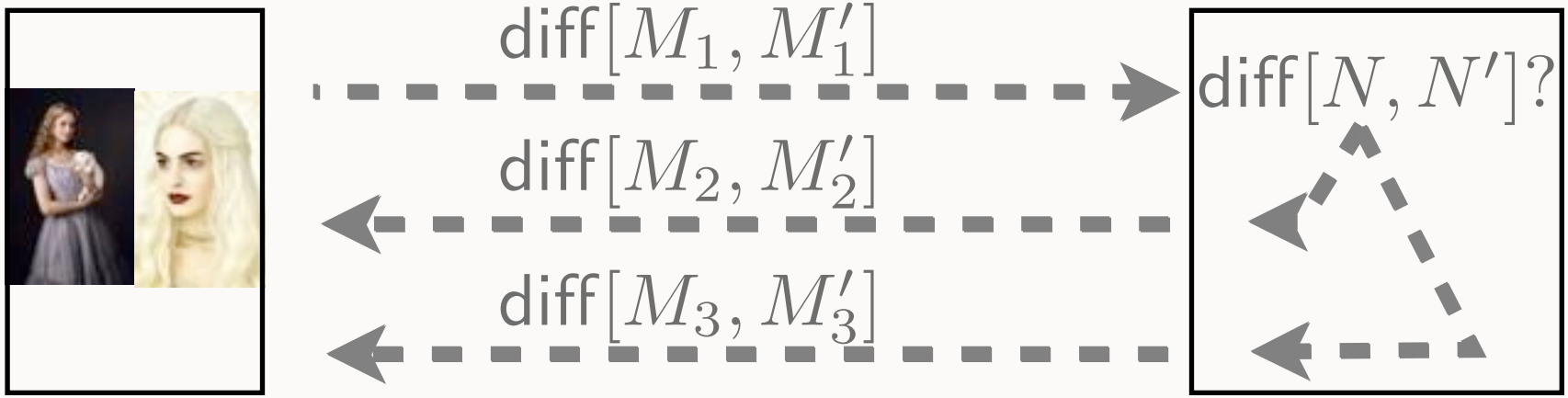


$$\rightarrow \mathrm{att}(M_1, M_1')$$

$$\mathrm{att}(N, N') \rightarrow \mathrm{att}(M_2, M_2')$$

CURRENT PROVERIF

- Biprocess



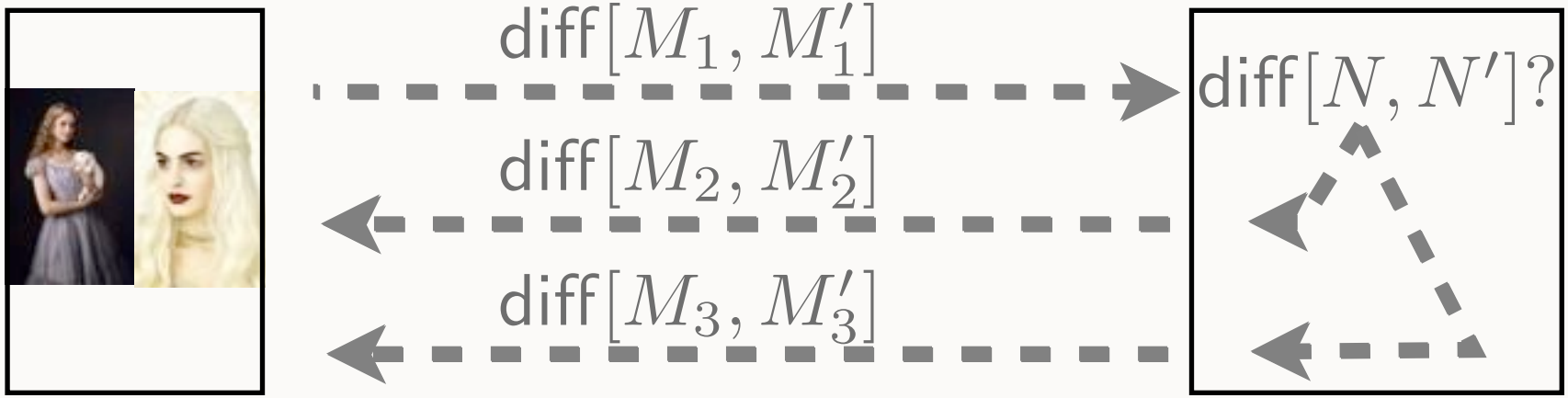
$$\rightarrow \mathrm{att}(M_1, M_1')$$

$$\mathrm{att}(N, N') \rightarrow \mathrm{att}(M_2, M_2')$$

$$\mathrm{att}(x, x') \wedge \forall y, y'.\ (x, x') \neq (N, N') \rightarrow \mathrm{att}(M_2, M_2')$$

CURRENT PROVERIF

- Biprocess



$$\to \mathrm{att}(M_1, M_1')$$

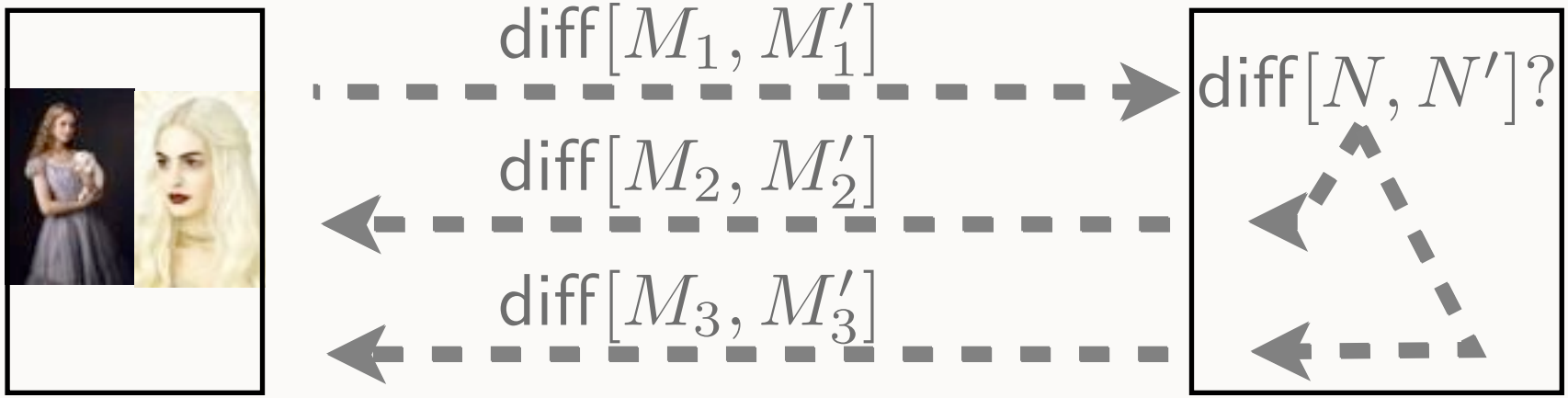
$$\mathrm{att}(N, N') \to \mathrm{att}(M_2, M_2')$$

$$\mathrm{att}(x, x') \wedge \forall y, y'.\ (x, x') \neq (N, N') \to \mathrm{att}(M_2, M_2')$$

$$\mathrm{att}(N, x') \wedge \forall y.\ x' \neq N' \to \mathrm{bad}$$

CURRENT PROVERIF

- Biprocess



$$\to \mathrm{att}(M_1, M_1')$$

$$\mathrm{att}(N, N') \to \mathrm{att}(M_2, M_2')$$

$$\mathrm{att}(x, x') \wedge \forall y, y'.\ (x, x') \neq (N, N') \to \mathrm{att}(M_2, M_2')$$

$$\mathrm{att}(N, x') \wedge \forall y.\ x' \neq N' \to \mathrm{bad}$$

$$\mathrm{att}(x, N') \wedge \forall y.\ x \neq N \to \mathrm{bad}$$

CURRENT PROVERIF

- Attacker clauses

$$\rightarrow \mathrm{att}(a, a) \qquad \text{with } a \text{ public}$$

$$\mathrm{att}(x, x') \wedge \mathrm{att}(y, y') \rightarrow \mathrm{att}(\{x\}_y, \{x'\}_{y'})$$

- New destructor reduction

$$\mathrm{att}(\{x\}_y, \{x'\}_{y'}) \wedge \mathrm{att}(y, y') \rightarrow \mathrm{att}(x, x')$$

- Destructor failure

$$\mathrm{att}(\{x\}_y, x') \wedge \mathrm{att}(y, y') \wedge \forall z.\ x' \neq \{z\}_{y'} \rightarrow \mathrm{bad}$$

$$\mathrm{att}(x, x') \wedge \mathrm{att}(x, y') \wedge x' \neq y' \rightarrow \mathrm{bad}$$

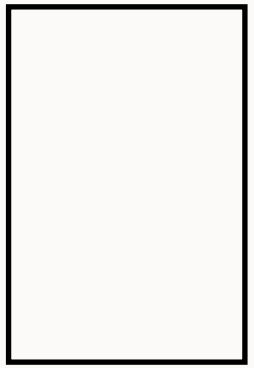
PROVERIF

A biprocess is *equivalent* if its horn clauses cannot derive *bad*

NEW PROVERIF

$\mathsf{diff}[\{x, y\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}]$

$\mathsf{ifthenelse}(\mathsf{diff}[y, y'], M_1, M_2, M_3)$



Intruder

Bob

$$M_1 = \mathsf{diff}[pk(k_A), pk(k_C)]$$
$$M_2 = \mathsf{diff}[\{x, N_b, pk(k_B)\}_y, \{x', N'_b, pk(k_B)\}_y]$$
$$M_3 = \mathsf{diff}[\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}$$

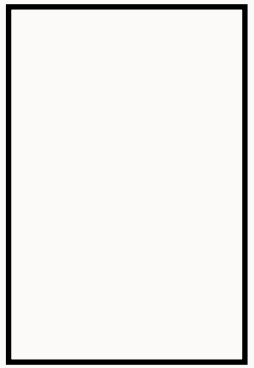
- Case 1 : Both tests succeed

$$\begin{array}{ll} & \mathrm{att}(\{x, pk(k_A)\}_{pk(k_B)}, \{x', pk(k_C)\}_{pk(k_B)}) \\ \rightarrow & \mathrm{att}(\{x, N_b, pk(k_B)\}_{pk(k_A)}, \{x', N'_b, pk(k_B)\}_{pk(k_C)}) \end{array}$$

NEW PROVERIF

$\mathsf{diff}[\{x, y\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}]$

$\mathsf{ifthenelse}(\mathsf{diff}[y, y'], M_1, M_2, M_3)$



Intruder

Bob

$$M_1 = \mathsf{diff}[pk(k_A), pk(k_C)]$$
$$M_2 = \mathsf{diff}[\{x, N_b, pk(k_B)\}_y, \{x', N'_b, pk(k_B)\}_y]$$
$$M_3 = \mathsf{diff}[\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}$$

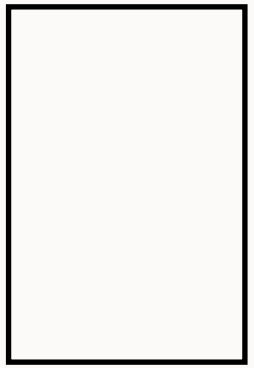
- Case 2 : Both tests fail

$$\begin{array}{ll} & \mathrm{att}(\{x, y'\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}) \\ \wedge & y \neq pk(k_A) \wedge y' \neq pk(k_C) \\ \rightarrow & \mathrm{att}(\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}) \end{array}$$

NEW PROVERIF

$\mathsf{diff}[\{x, y\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}]$

$\mathsf{ifthenelse}(\mathsf{diff}[y, y'], M_1, M_2, M_3)$



Intruder

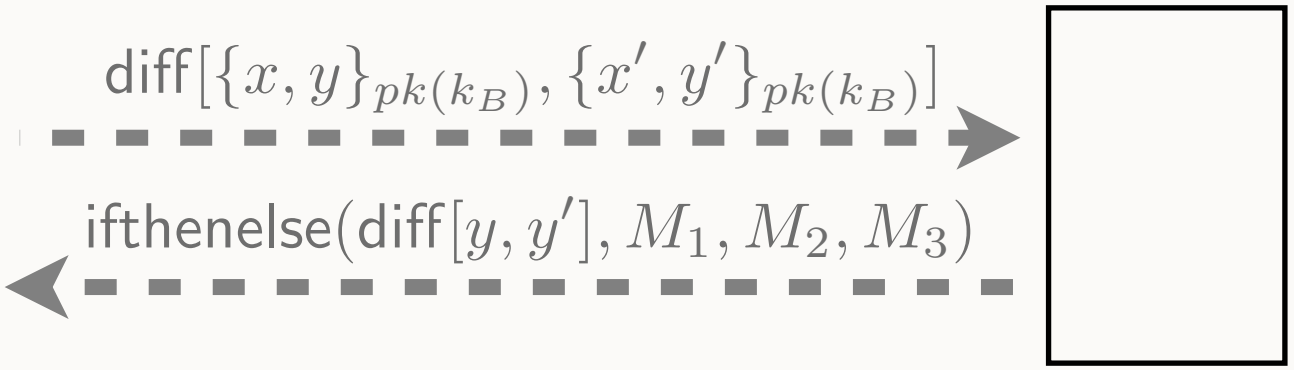
Bob

$$M_1 = \mathsf{diff}[pk(k_A), pk(k_C)]$$
$$M_2 = \mathsf{diff}[\{x, N_b, pk(k_B)\}_y, \{x', N'_b, pk(k_B)\}_y]$$
$$M_3 = \mathsf{diff}[\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}$$

- Old case 3 : Left test succeeds, right test fails

$$\begin{array}{rl} & \mathrm{att}(\{x, pk(k_A)\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}) \\ \wedge & y' \neq pk(k_C) \\ \rightarrow & \mathrm{bad} \end{array}$$

NEW PROVERIF



Intruder

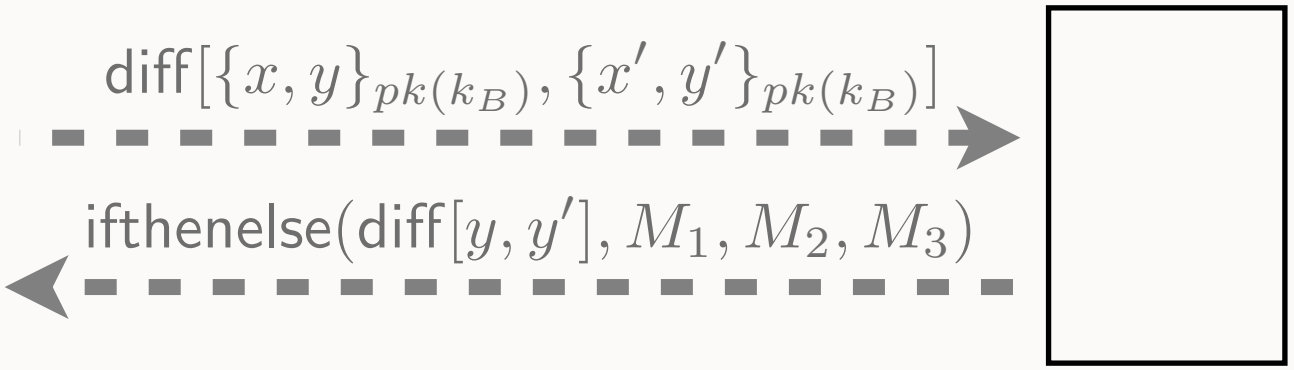
Bob

$$M_1 = \mathsf{diff}[pk(k_A), pk(k_C)]$$
$$M_2 = \mathsf{diff}[\{x, N_b, pk(k_B)\}_y, \{x', N'_b, pk(k_B)\}_y]$$
$$M_3 = \mathsf{diff}[\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}$$

- New case 3 : Left test succeeds, right test fails

$$\begin{array}{rl} & \mathrm{att}(\{x, pk(k_A)\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}) \\ \wedge & y' \neq pk(k_C) \\ \rightarrow & \mathrm{att}(\{x, N_b, pk(k_B)\}_{pk(k_A)}, \{N'\}_{pk(k_C)}) \end{array}$$

NEW PROVERIF



Intruder

Bob

$$M_1 = \mathsf{diff}[pk(k_A), pk(k_C)]$$
$$M_2 = \mathsf{diff}[\{x, N_b, pk(k_B)\}_y, \{x', N'_b, pk(k_B)\}_y]$$
$$M_3 = \mathsf{diff}[\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}$$

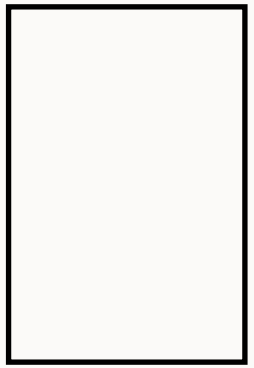
- Old case 4 : Left test fails, right test succeeds

$$\begin{array}{rl} & \mathrm{att}(\{x,y\}_{pk(k_B)}, \{x', pk(k_C)\}_{pk(k_B)}) \\ \wedge & y \neq pk(k_A) \\ \rightarrow & \text{bad} \end{array}$$

NEW PROVERIF

$\mathsf{diff}[\{x, y\}_{pk(k_B)}, \{x', y'\}_{pk(k_B)}]$

$\mathsf{ifthenelse}(\mathsf{diff}[y, y'], M_1, M_2, M_3)$



Intruder

Bob

$$M_1 = \mathsf{diff}[pk(k_A), pk(k_C)]$$
$$M_2 = \mathsf{diff}[\{x, N_b, pk(k_B)\}_y, \{x', N'_b, pk(k_B)\}_y]$$
$$M_3 = \mathsf{diff}[\{N\}_{pk(k_A)}, \{N'\}_{pk(k_C)}$$

- New case 4 : Left test fails, right test succeeds

$$\begin{array}{rl} & \mathrm{att}(\{x, y\}_{pk(k_B)}, \{x', pk(k_C)\}_{pk(k_B)}) \\ \wedge & y \neq pk(k_A) \\ \rightarrow & \mathrm{att}(\{N\}_{pk(k_A)}, \{x', N'_b, pk(k_B)\}_{pk(k_C)}) \end{array}$$

CONCLUSION

- Contribution

We introduce destructor with tests between terms

$$\mathrm{g}(M_1, \ldots, M_n) \rightarrow M \text{ with } \Phi$$

+ Prove equivalence on larger class of protocol (e.g. private authentication)
+ Can perform automatic detection and transformation of protocol
+ Could be used for reachability properties
- E-passport protocol is still not handle by ProVerif

- Future work
  - Remove even further false attack due to parameter of nonces
  - E-passport protocol ?