Foundations of Security

in Distributed Systems

Joshua D. Guttman
Worcester Polytechnic Institute
The MITRE Corporation

March 2013
Bertinoro International Spring School

Thanks to the US National Science Foundation, under grant 1116557

guttman@wpi.edu

Goals of this Course

- Study the main mechanisms for
  - authentication
  - confidentiality
  - authorization
- Develop methods for
  - finding attacks
  - proof
  - analysis
  - systematic design
- Present a useful tool, CPSA
- Work within a single point of view mainly, the strand space framework

Structure of the Lectures

Monday	Breaking and proving protocols
Tuesday	Shapes and CPSA; the anatomy of TLS
Wednesday	Protocol goals, composition, separability, transformation
Thursday	Authorization, trust management, state change
Friday	Foundations of cryptography, related to strands

Schedule at
`http://cs.wpi.edu/~guttman/biss/schedule.pdf`

How to Break a Protocol

or prove it correct

Joshua D. Guttman

March 2013

Bertinoro International Spring School

Thanks to the US National Science Foundation, under grant 1116557

guttman@wpi.edu

What is a Security Protocol?

- For instance, SSL (= TLS), SSH, OAuth, IKE
  - Short sequence of messages
  - Use cryptography for authentication, confidentiality
  - Provide reliable communication
  - Despite malicious attackers, who may control network

What is a Security Protocol?

- For instance, SSL (= TLS), SSH, OAuth, IKE
  - Short sequence of messages
  - Use cryptography for authentication, confidentiality
  - Provide reliable communication
  - Despite malicious attackers, who may control network
- Protocols establish trust
  - Remote access
  - Secure networking
  - E-commerce

What is a Security Protocol?

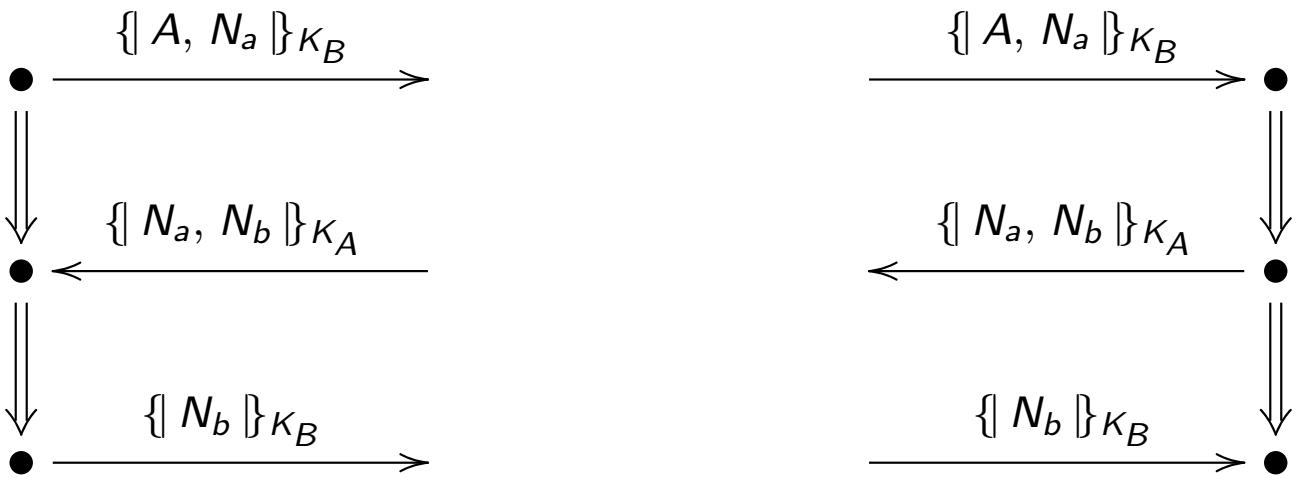
- For instance, SSL (= TLS), SSH, OAuth, IKE
  - Short sequence of messages
  - Use cryptography for authentication, confidentiality
  - Provide reliable communication
  - Despite malicious attackers, who may control network
- Protocols establish trust
  - Remote access
  - Secure networking
  - E-commerce
- Security protocols are often wrong
  - Active attackers may subvert goals
  - Even if cryptography is perfect
  - Often fail because assumptions unclear
  - Often misapplied because goals unclear

How to break a protocol

- Try to prove it correct
  - Where you get stuck:
    that's where the flaw is
- Central focus: services provided by protocol
  - Actions the protocol requires
    compliant principals to perform
  - Produce values useful to adversary
- Each protocol poses certain challenges to attacker
  - Services help adversary
    to meet those challenges

Needham-Schroeder Protocol

A Simplest Example

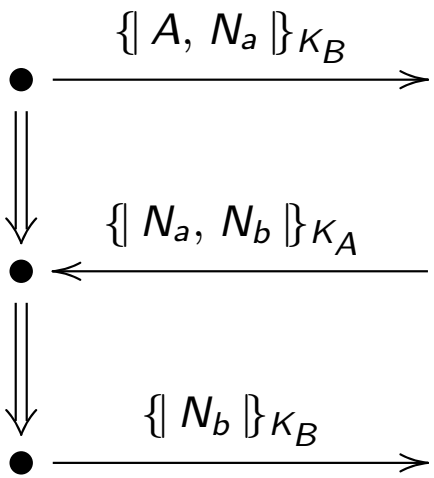


Terminology:

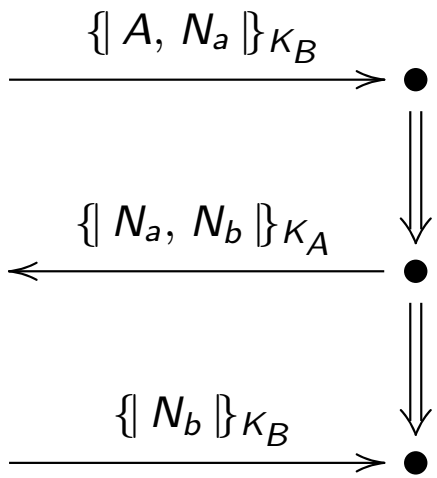
K_A, K_B	Public encryption keys for A and B
N_a, N_b	Randomly chosen bitstrings "nonces"
$\{\mid t \mid\}_K$	t encrypted using K
$N_a,\ N_b$	new shared secret

Roles are templates

Variables instantiated by values



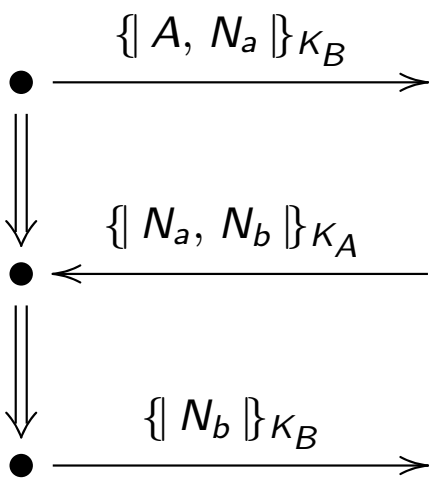
Initiator



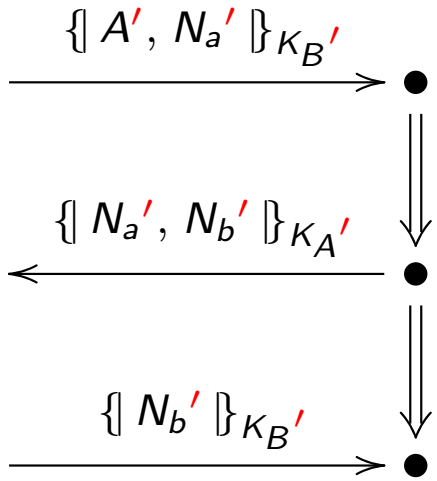
Responder

Roles are templates

Variables instantiated by values



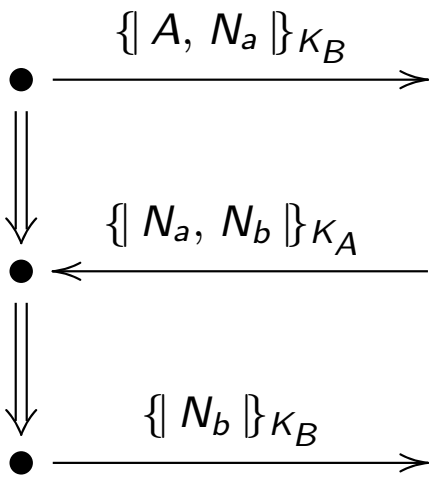
Initiator



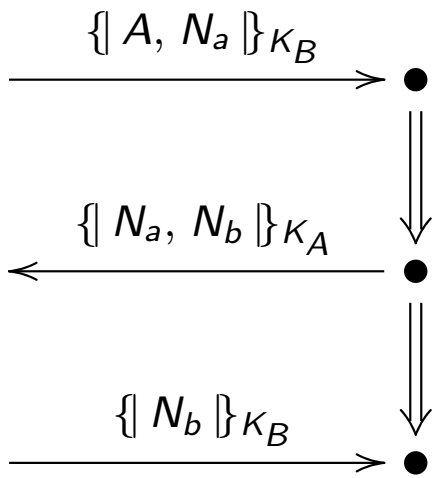
Responder

Roles are templates

Variables instantiated by values



Initiator
$[A, B, N_a, N_b]$



Responder
$[A, B, N_a, N_b]$

Essence of Cryptography

for today's lecture

- Symmetric key crypto: single shared key for sender, receiver
  - Same key makes ciphertext, extracts plaintext

Essence of Cryptography

for today's lecture

- Symmetric key crypto: single shared key for sender, receiver
  - Same key makes ciphertext, extracts plaintext
- Public key crypto: two related keys, one private, the other public
  - Encryption: Public key creates, private key decrypts
  - Signature: Private key creates, public key verifies

Essence of Cryptography

for today's lecture

- Symmetric key crypto: single shared key for sender, receiver
  - Same key makes ciphertext, extracts plaintext
- Public key crypto: two related keys, one private, the other public
  - Encryption: Public key creates, private key decrypts
  - Signature: Private key creates, public key verifies
- Terminology: A's public key: K_A A's private key: K_A^{-1}

In symmetric crypto, $K = K^{-1}$

Essence of Cryptography

for today's lecture

- Symmetric key crypto: single shared key for sender, receiver
  - Same key makes ciphertext, extracts plaintext
- Public key crypto: two related keys, one private, the other public
  - Encryption: Public key creates, private key decrypts
  - Signature: Private key creates, public key verifies
- Terminology: A's public key: K_A A's private key: K_A^{-1}

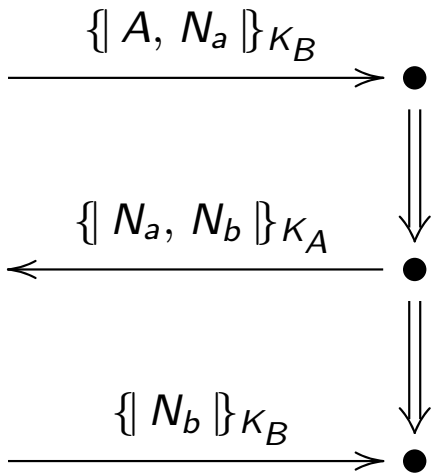
$$\text{In symmetric crypto, } K = K^{-1}$$

- Uncompromised key:

Key used only in accordance with this protocol

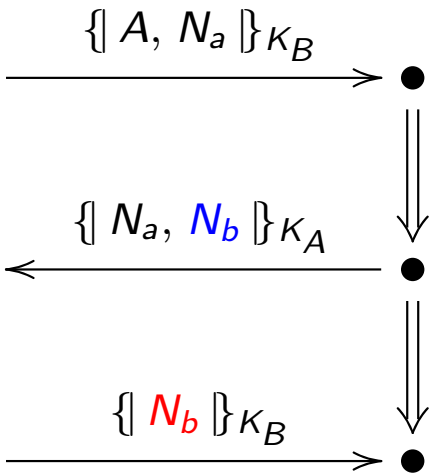
Needham-Schroeder: How does it work?

K_A^{-1} uncompromised, N_b fresh



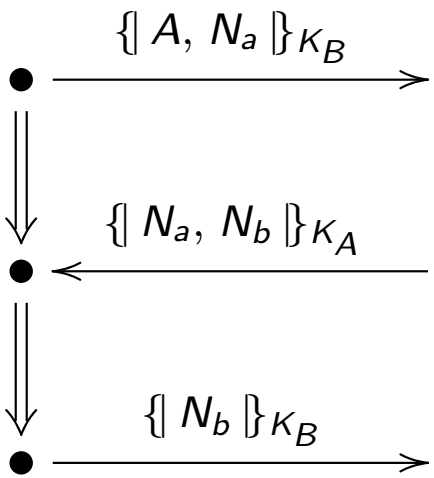
Needham-Schroeder: How does it work?

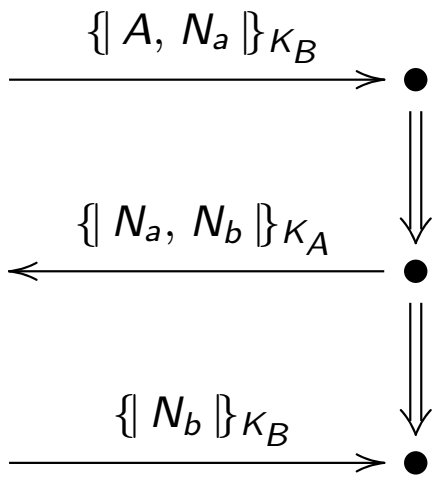
K_A^{-1} uncompromised, N_b fresh



Roles are templates

Variables instantiated by values





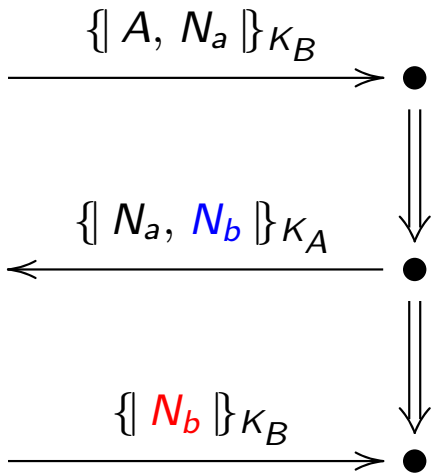
Initiator
$[A, B, N_a, N_b]$

Responder
$[A, B, N_a, N_b]$

whitespace

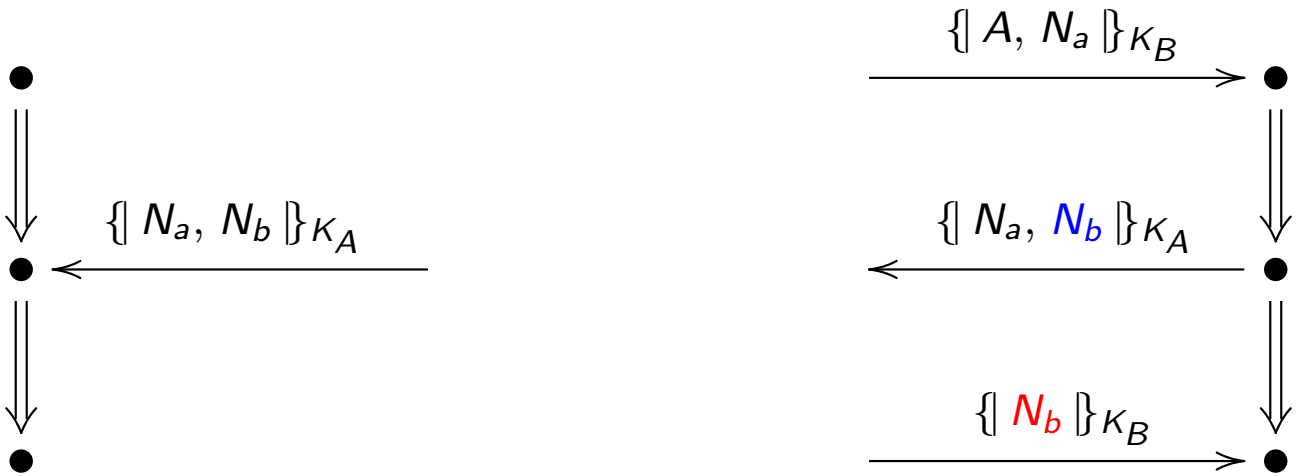
Needham-Schroeder: How does it work?

K_A^{-1} uncompromised, N_b fresh



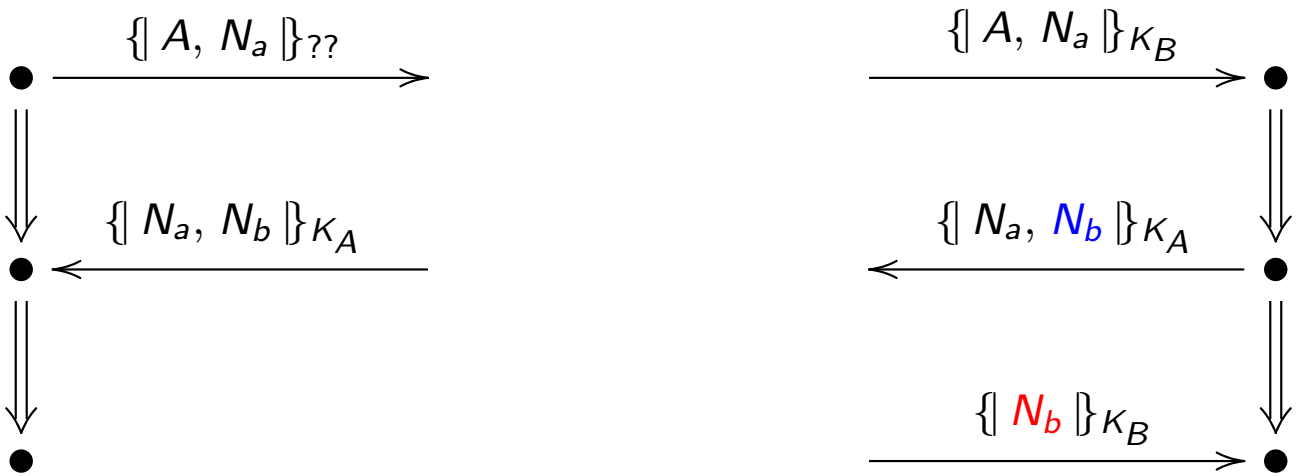
Needham-Schroeder: How does it work?

K_A^{-1} uncompromised, N_b fresh



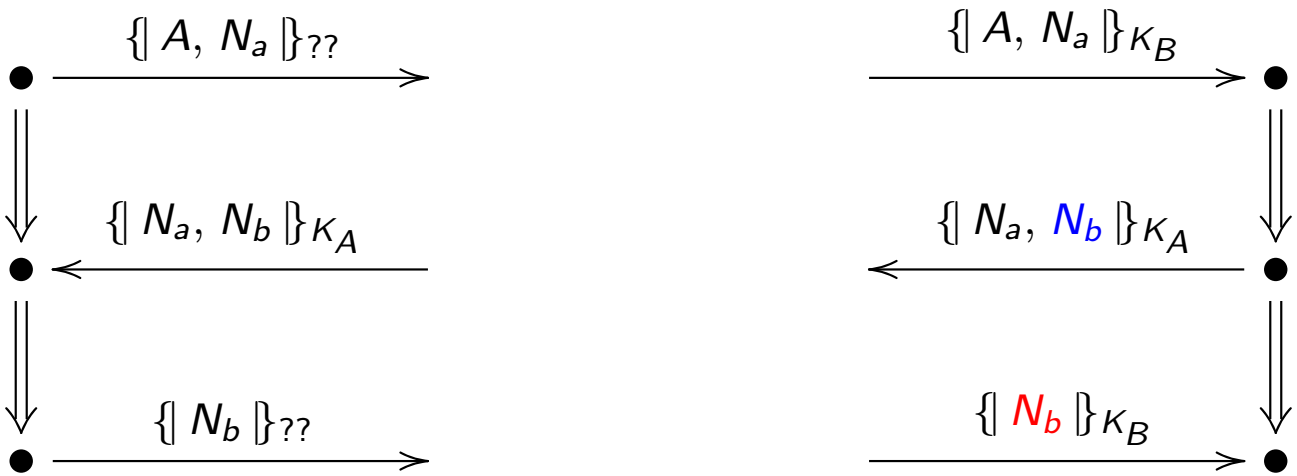
Needham-Schroeder: How does it work?

K_A^{-1} uncompromised, N_b fresh



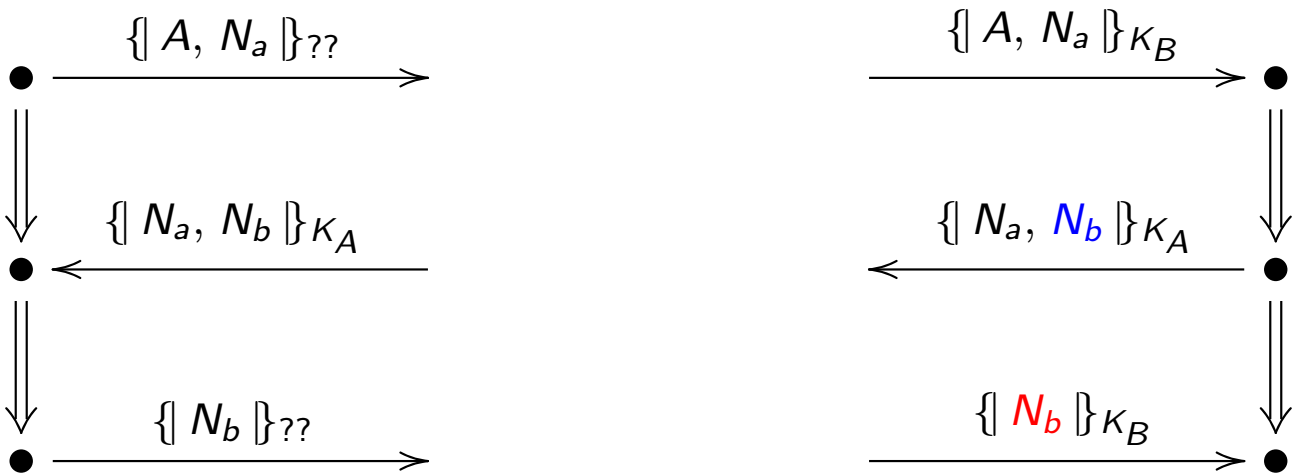
Needham-Schroeder: How does it work?

K_A^{-1} uncompromised, N_b fresh



Needham-Schroeder: How does it work?

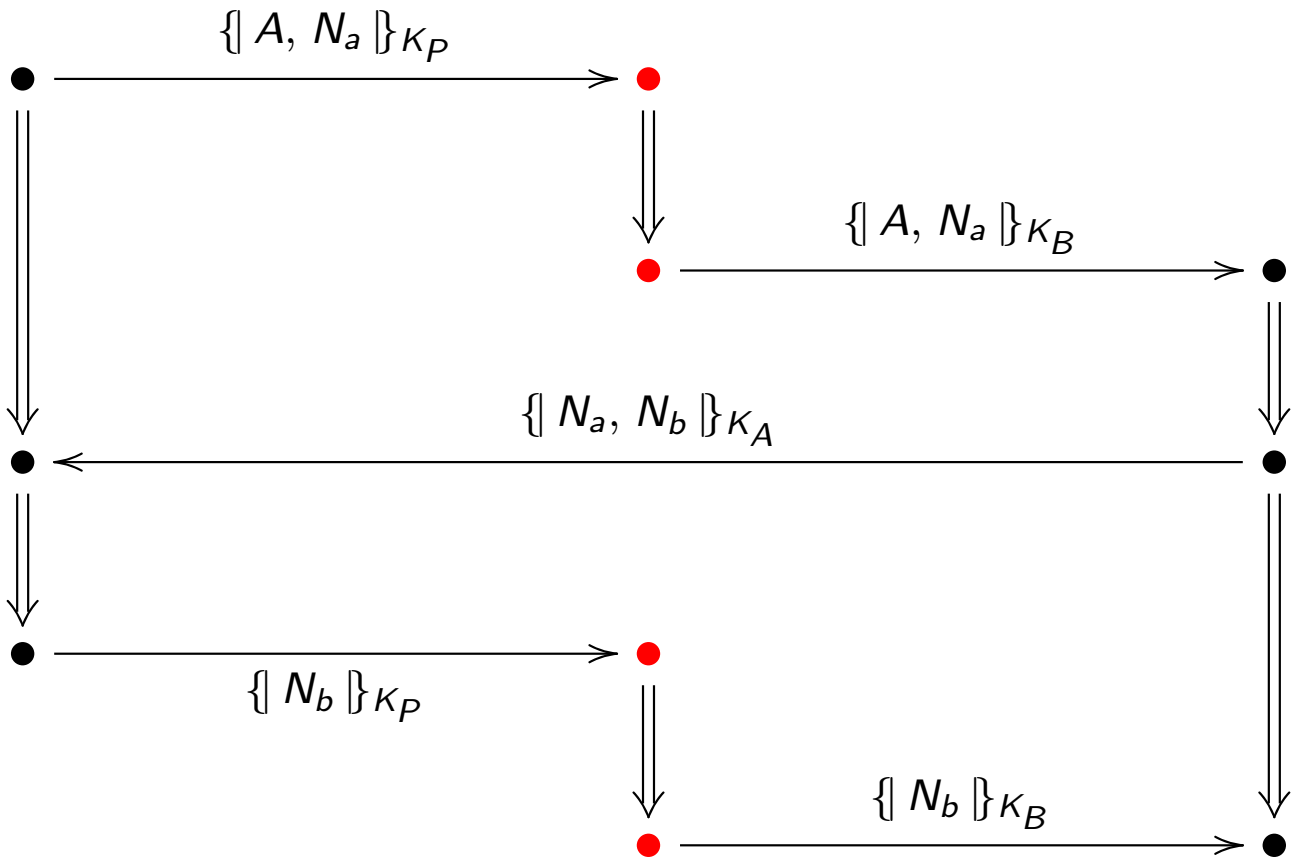
K_A^{-1} uncompromised, N_b fresh



Whoops

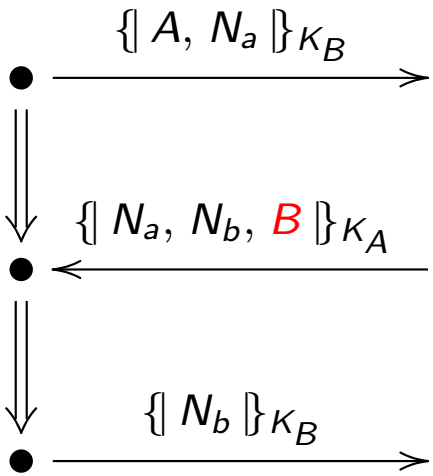
Needham-Schroeder Failure

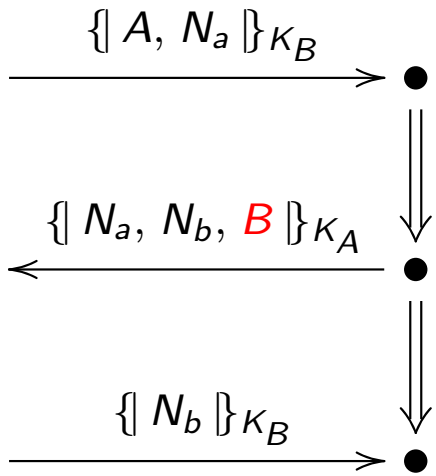
If $?? = K_P$, where K_P^{-1} is compromised:



due to Gavin Lowe

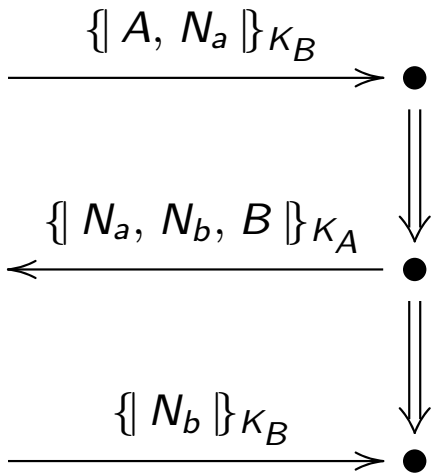
Needham-Schroeder-Lowe





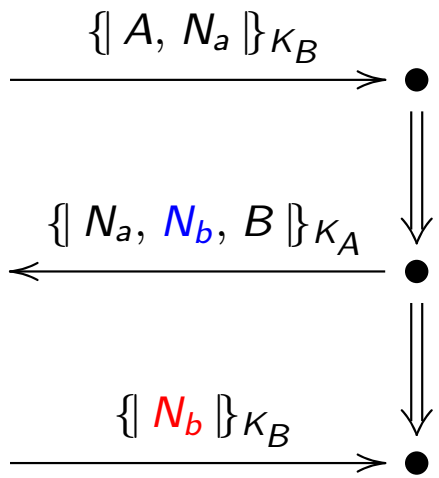
Needham-Schroeder-Lowe Analysis

K_A^{-1} uncompromised, N_b fresh

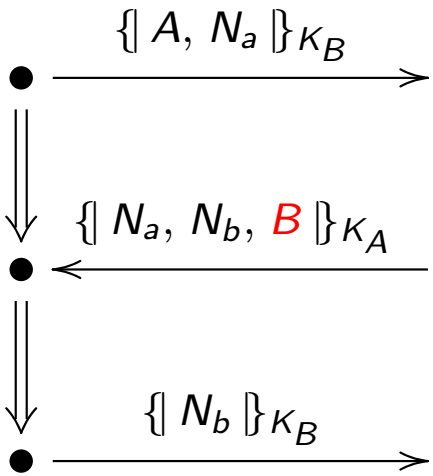


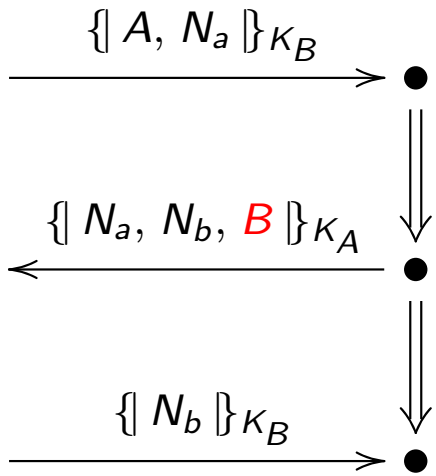
Needham-Schroeder-Lowe Analysis

K_A^{-1} uncompromised, N_b fresh



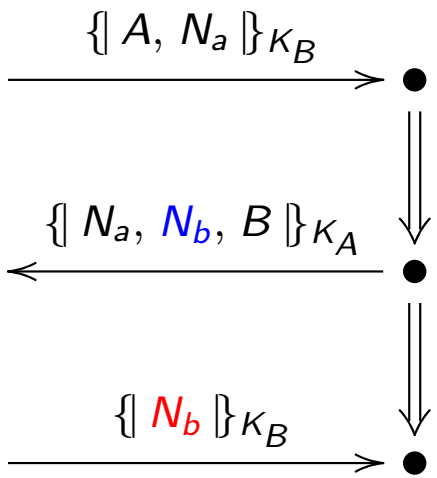
Needham-Schroeder-Lowe





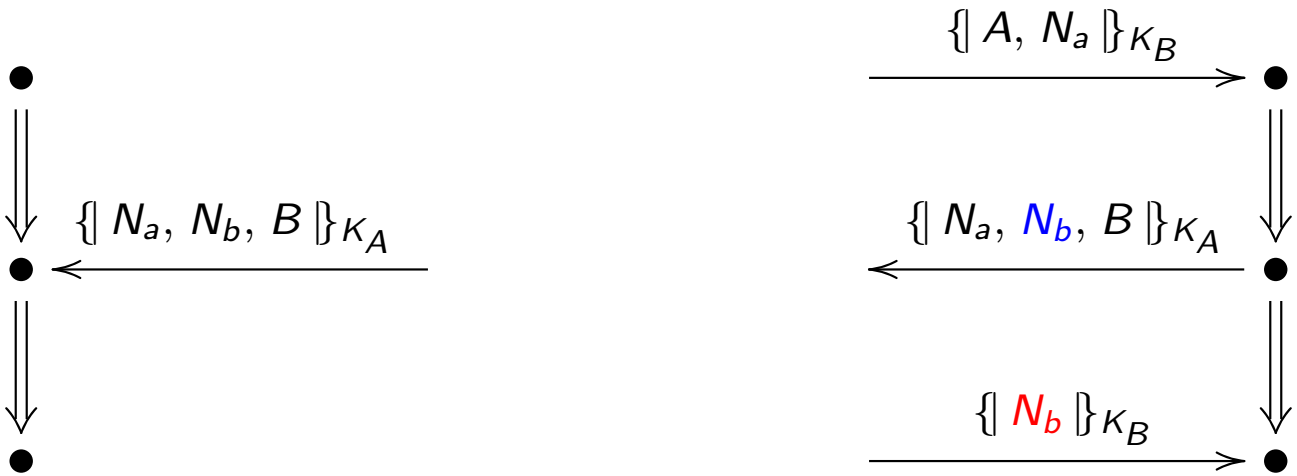
Needham-Schroeder-Lowe Analysis

K_A^{-1} uncompromised, N_b fresh



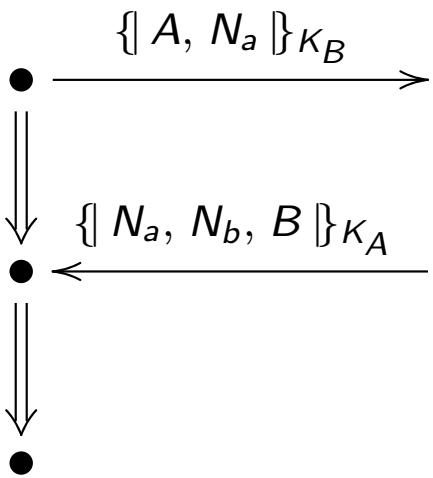
Needham-Schroeder-Lowe Analysis

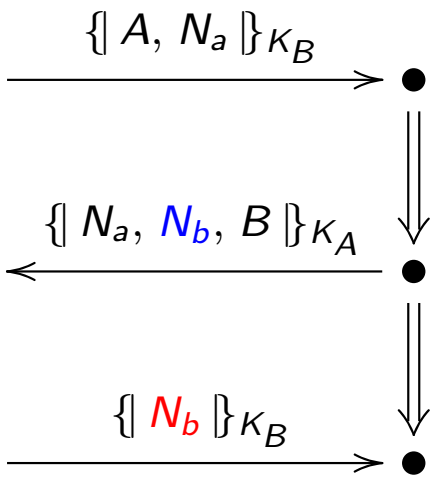
K_A^{-1} uncompromised, N_b fresh



Needham-Schroeder-Lowe Analysis

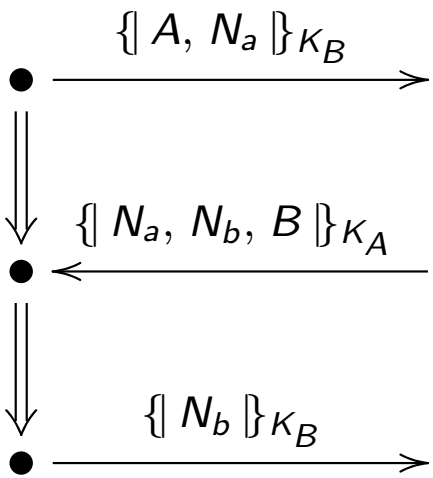
K_A^{-1} uncompromised, N_b fresh

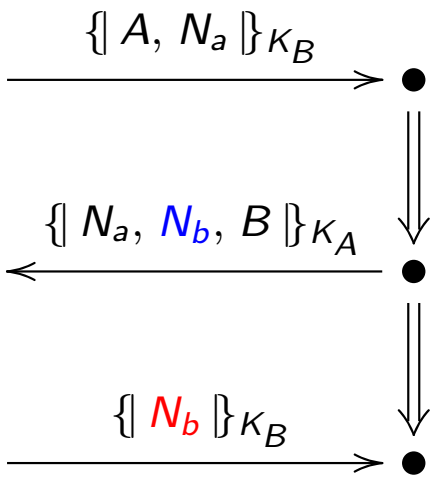




Needham-Schroeder-Lowe Analysis

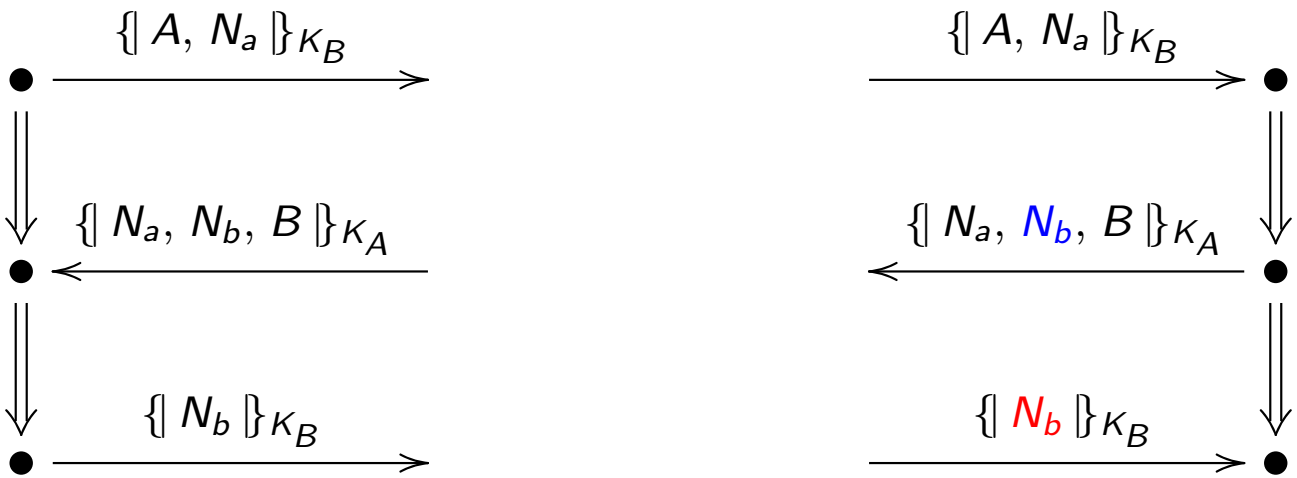
K_A^{-1} uncompromised, N_b fresh





Needham-Schroeder-Lowe Analysis

K_A^{-1} uncompromised, N_b fresh

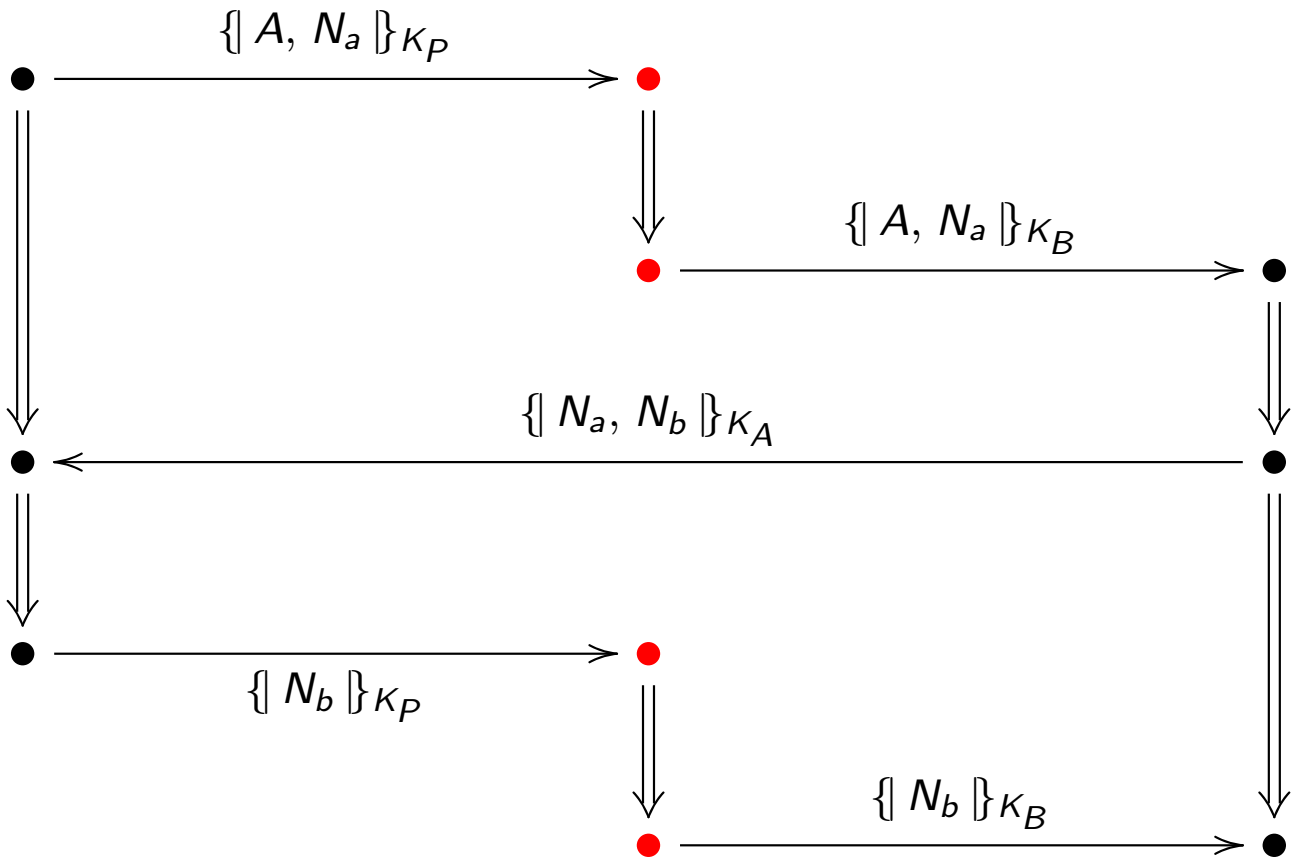


This proves "everything that's true" in this situation

How to break a protocol: Unintended services

Needham-Schroeder Failure

If $?? = K_P$, where K_P^{-1} is compromised:



due to Gavin Lowe

Diagnosis of a failure

- Who did the protocol betray? Not A, who
  - Meant to share nonces with P
  - Wrongly assumed K_P^{-1} uncompromised

Diagnosis of a failure

- Who did the protocol betray? Not A, who
  - Meant to share nonces with P
  - Wrongly assumed K_P^{-1} uncompromised
- B was betrayed
  - Meant to share nonces with A
  - Rightly assumed K_A^{-1} uncompromised
  - Secrecy failed, since P knows nonces
  - Authentication failed: A had no run with B

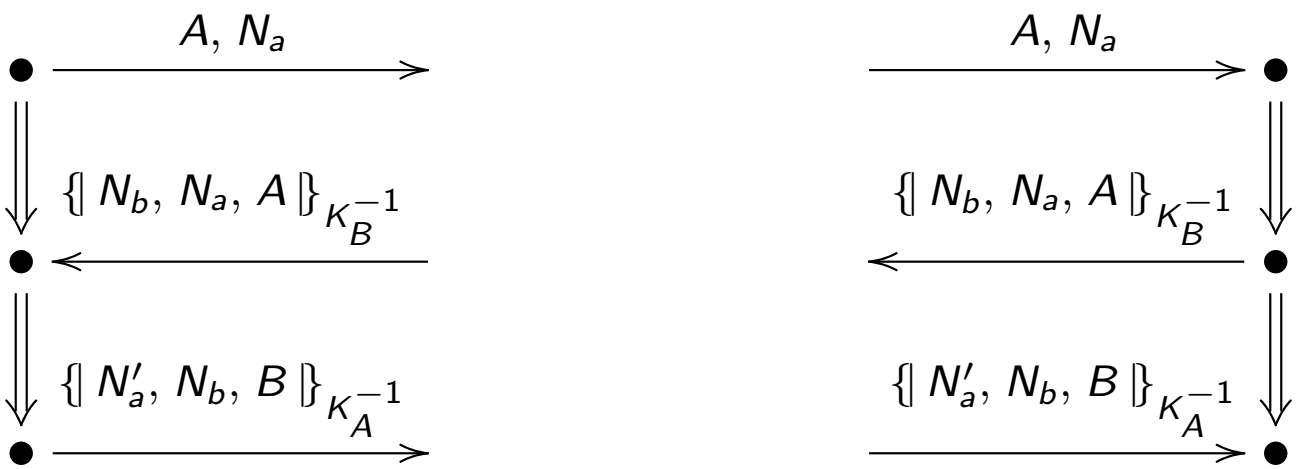
Diagnosis of a failure

- Who did the protocol betray? Not A, who
  - Meant to share nonces with P
  - Wrongly assumed K_P^{-1} uncompromised
- B was betrayed
  - Meant to share nonces with A
  - Rightly assumed K_A^{-1} uncompromised
  - Secrecy failed, since P knows nonces
  - Authentication failed: A had no run with B
- How? A offered P a service:
  - Gave P a nonce N_A
  - Promised to translate any $\{\!|\, N_a,\ N' \,|\!\}_{K_A}$ to $\{\!|\, N' \,|\!\}_{K_P}$
  - "Unintended service"

Diagnosis of a failure

- Who did the protocol betray? Not A, who
  - Meant to share nonces with P
  - Wrongly assumed K_P^{-1} uncompromised
- B was betrayed
  - Meant to share nonces with A
  - Rightly assumed K_A^{-1} uncompromised
  - Secrecy failed, since P knows nonces
  - Authentication failed: A had no run with B
- How? A offered P a service:
  - Gave P a nonce N_A
  - Promised to translate any $\{\!|\, N_a,\ N' \,|\!\}_{K_A}$ to $\{\!|\, N' \,|\!\}_{K_P}$
  - "Unintended service"
- Attacker needs to get some value i.e. N_b
  Regular participant unwittingly provides it

Another Example: ISO reject



Digital signatures provide authentication

No new secrets

Diagnosis: ISO reject

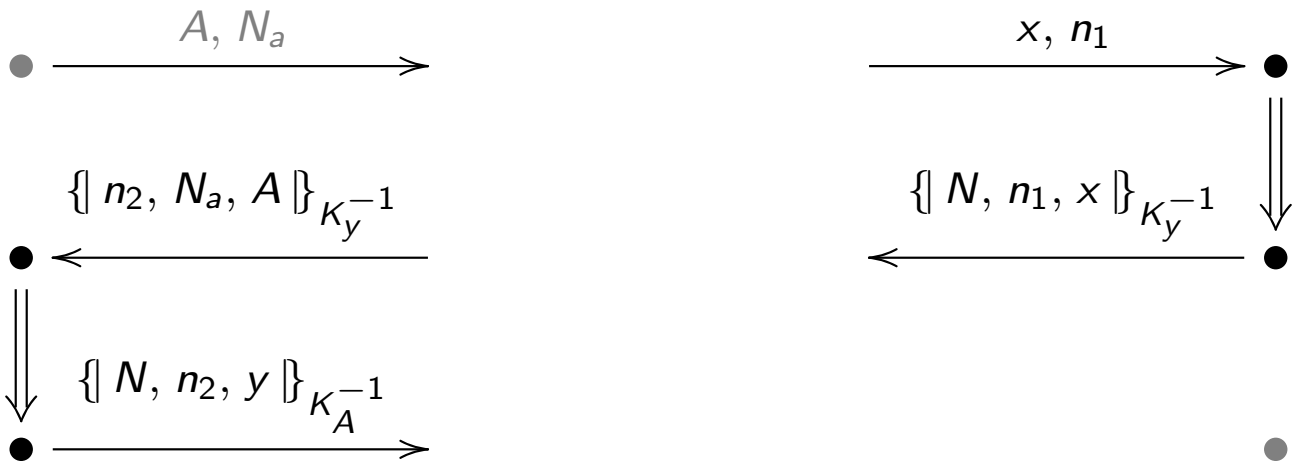
- Responder only gets two messages
  - First message $A,\ N_a$ has no authenticating force
  - Anybody can create a message like that
  - "Junk term"
- Attacker only needs to create

$$\{\!|\ N'_a,\ N_b,\ B\ |\!\}_{K_A^{-1}}$$

  for some N'_a
- What services are useful?

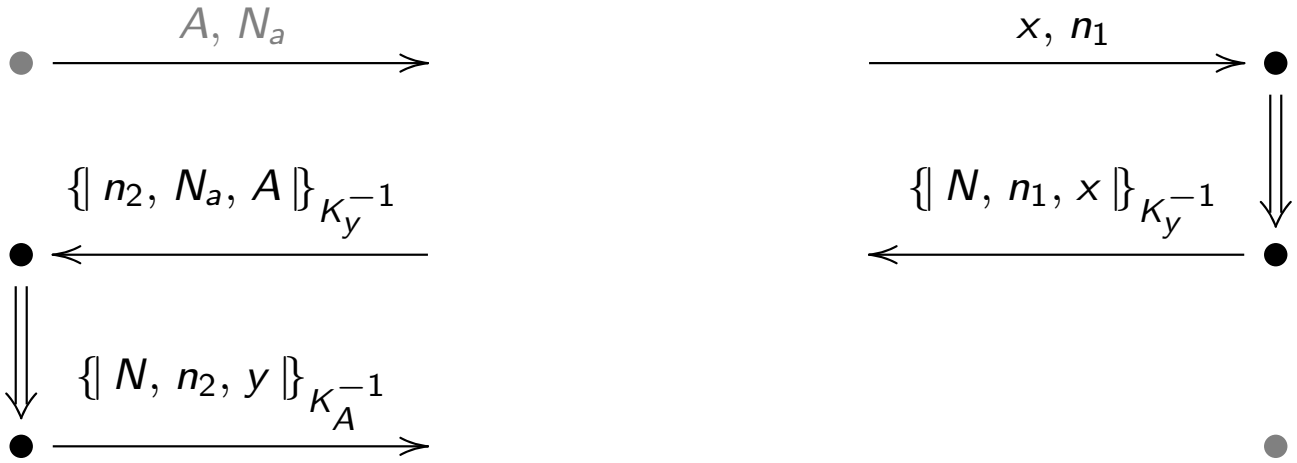
Services in ISO reject

lower case letters are variables chosen by adversary



Services in ISO reject

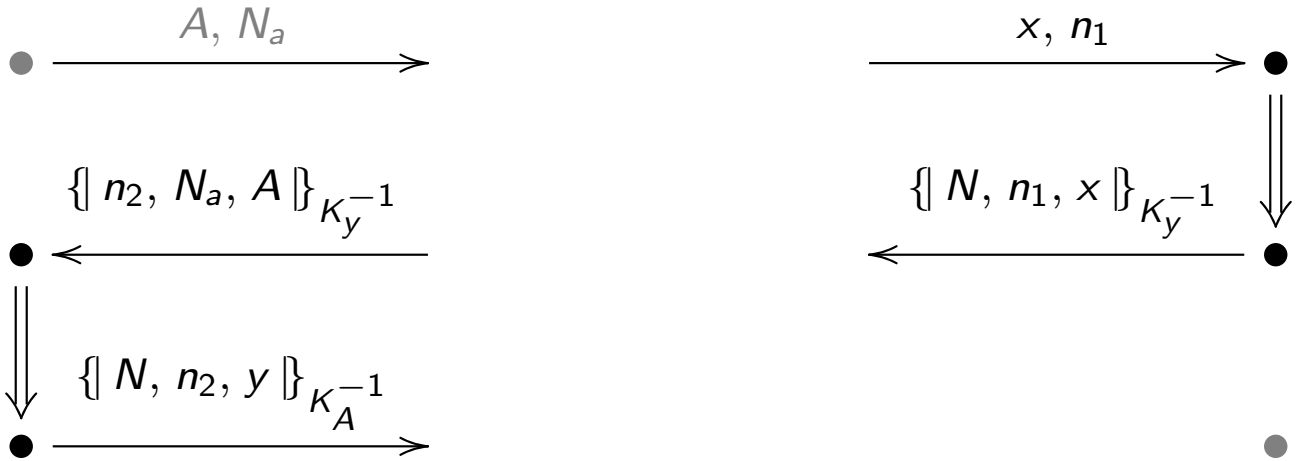
lower case letters are variables chosen by adversary



- Adversary freely instantiates lower-case variables

Services in ISO reject

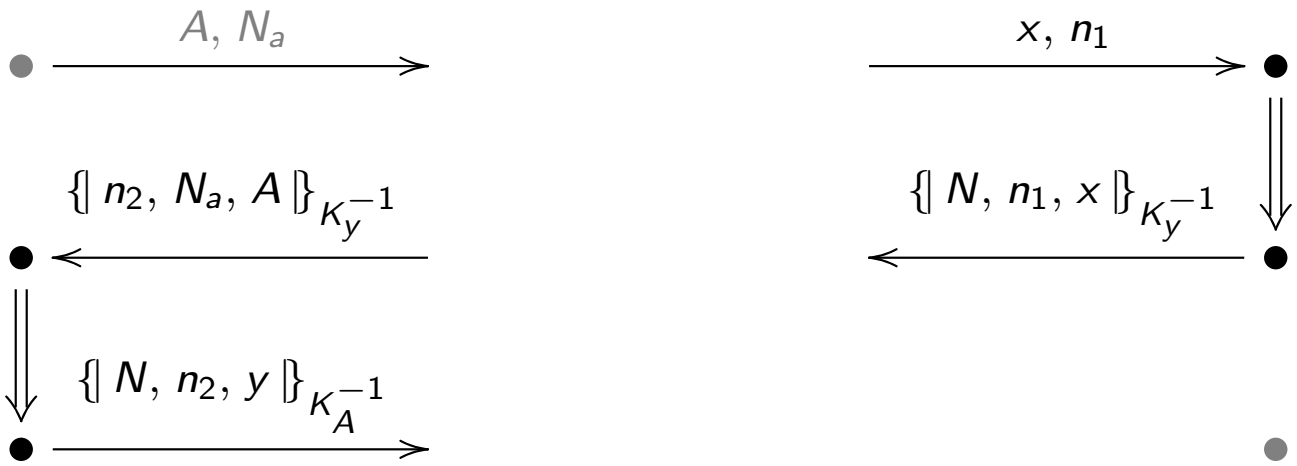
lower case letters are variables chosen by adversary



- Adversary freely instantiates lower-case variables
- Want to produce $\{\!|\ N,\ N_b,\ B\ |\!\}_{K_A^{-1}}$ for some N

Services in ISO reject

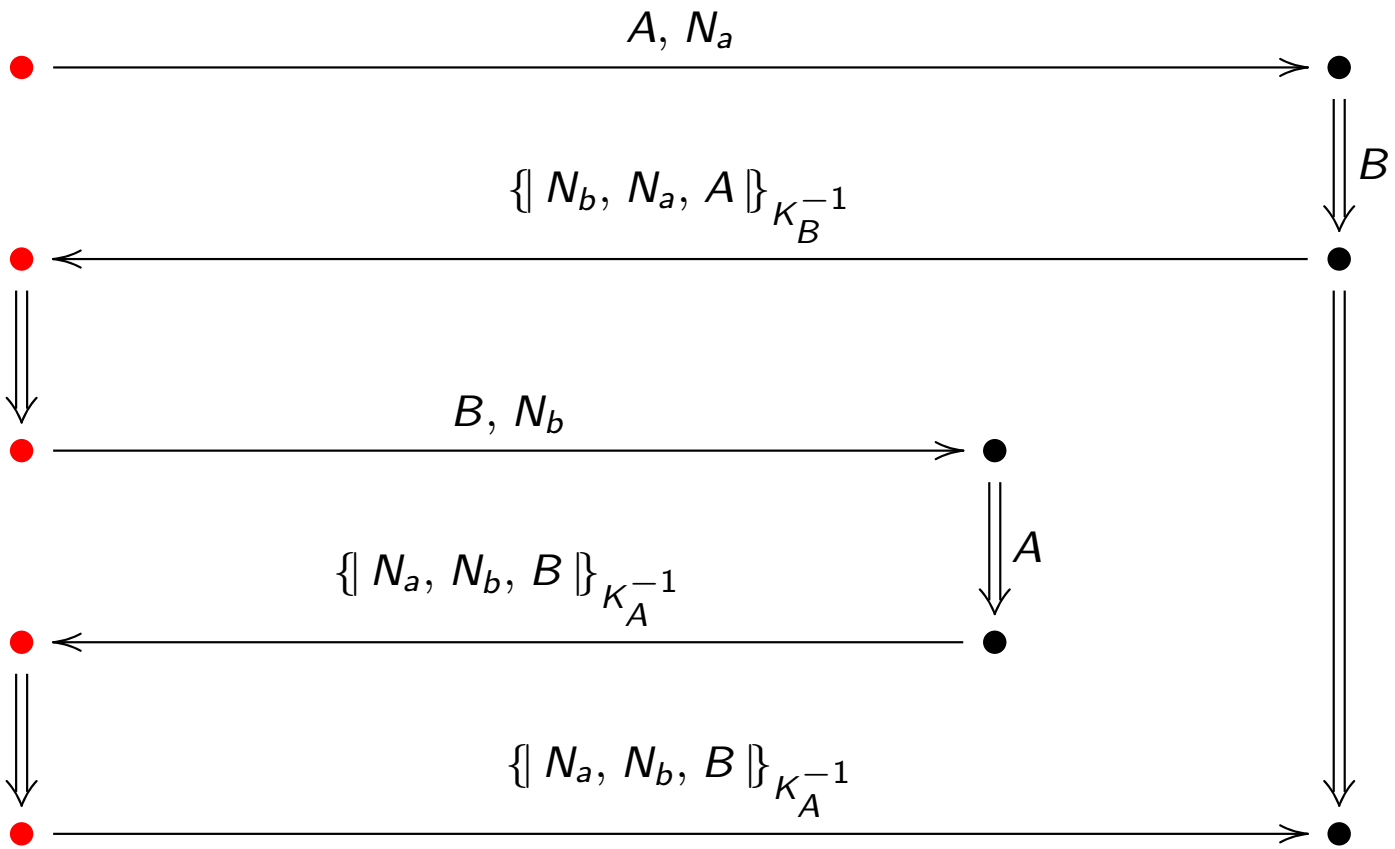
lower case letters are variables chosen by adversary



- Adversary freely instantiates lower-case variables
- Want to produce $\{\!|\ N,\ N_b,\ B\ |\!\}_{K_A^{-1}}$ for some N
- Can use A as respondent, $B,\ N_b$ as selected msg i.e. substitution $[B/x, A/y, N_b/n_1]$

The resulting attack

The Canadian Attack



What goal is refuted?

- A executed a signature
- So "entity authentication" may hold for A

Whatever that means

- But A never initiated any session with B

Dolev-Yao Attacks: A Recipe

- Identify and discard "junk" messages
  - They don't contribute to authentication
  - Remaining (non-junk) incoming messages "Challenge"
  - Adversary needs to synthesize them

Dolev-Yao Attacks: A Recipe

- Identify and discard "junk" messages
  - They don't contribute to authentication
  - Remaining (non-junk) incoming messages "Challenge"
  - Adversary needs to synthesize them
- Look for unintended services
  - Criterion: Can they build challenge messages?

Dolev-Yao Attacks: A Recipe

- Identify and discard "junk" messages
  - They don't contribute to authentication
  - Remaining (non-junk) incoming messages "Challenge"
  - Adversary needs to synthesize them
- Look for unintended services
  - Criterion: Can they build challenge messages?
- Combine unintended services

What unintended services occur?

Signature:	N_a	$\mapsto$	$\{ N_a \}_{K^{-1}}$
Encryption:	N_a	$\mapsto$	$\{ N_a \}_K$
Decryption:	$\{ N_a \}_K$	$\mapsto$	N_a
Translation:	$\{ N_a \}_K$	$\mapsto$	$\{ N_a \}_{K'}$

Examples:

Signature service: *ISO reject protocol*
Encryption service: *Woo-Lam*
Decryption service: *None* *(too obvious?)*
Key-translation service: *NS PK*

The Dolev-Yao Problem

- Given a protocol, and assuming all cryptography perfect, find
  - What secrecy properties
  - What authentication properties

  the protocol achieves
- Find counterexamples to other properties
  - Unintended services useful
- What does perfect cryptography mean?
  - No collisions
  - Need key to make encrypted value
  - Need key to decrypt and recover plaintext

How to break a protocol

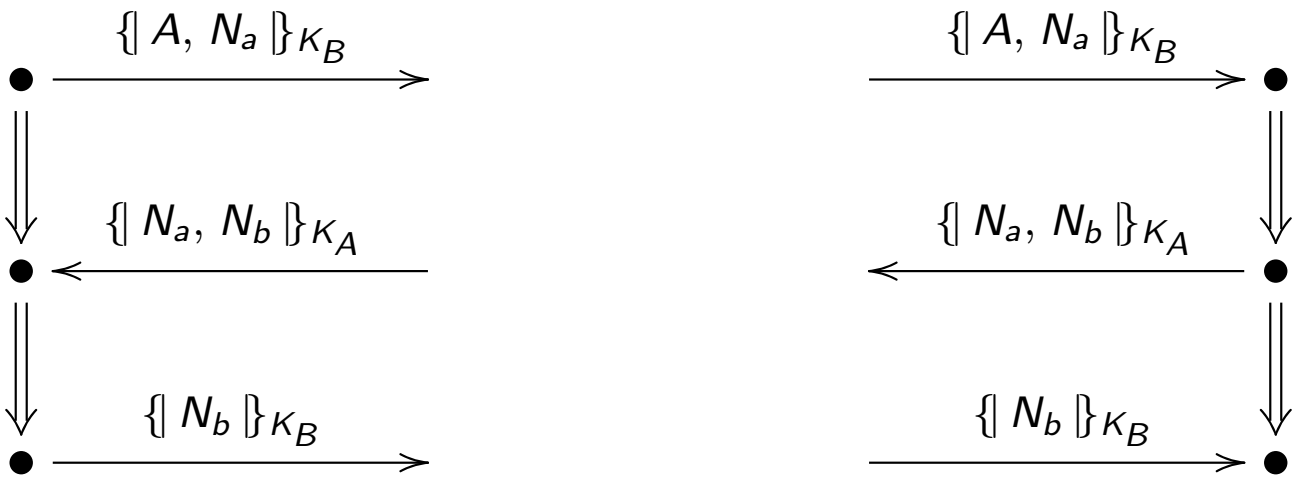
- Try to prove it correct
  - Where you get stuck:
    that's where the flaw is
- Central focus: services provided by protocol
  - Actions the protocol requires
    compliant principals to perform
  - Produce values useful to adversary
- Each protocol poses certain challenges to attacker
  - Services help adversary
    to meet those challenges

How to break a protocol

- Try to prove it correct
  - Where you get stuck:
    that's where the flaw is
- Central focus: services provided by protocol
  - Actions the protocol requires
    compliant principals to perform
  - Produce values useful to adversary
- Each protocol poses certain challenges to attacker
  - Services help adversary
    to meet those challenges
- How to prove a protocol:
  - Try to break it
  - If you get stuck — run out of services — it's correct

Needham-Schroeder Protocol

A Simplest Example

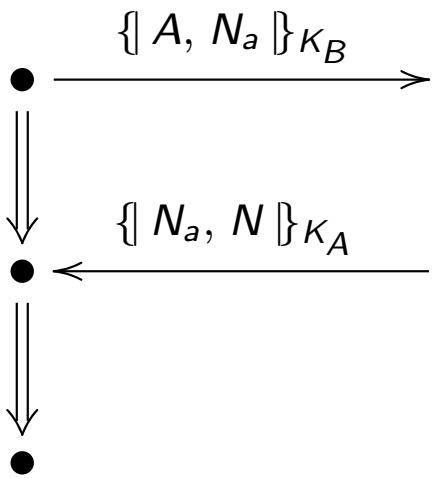


Terminology:

K_A, K_B	Public encryption keys for A and B
N_a, N_b	Randomly chosen bitstrings "nonces"
$\{\mid t \mid\}_K$	t encrypted using K
$N_a,\ N_b$	new shared secret

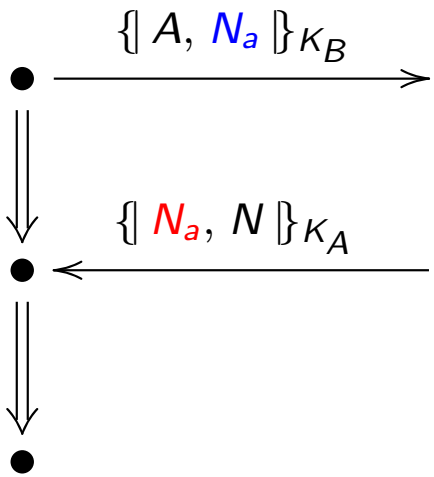
NS Analysis: Initiator's point of view

Assume K_A^{-1}, K_B^{-1} uncompromised, N_a fresh



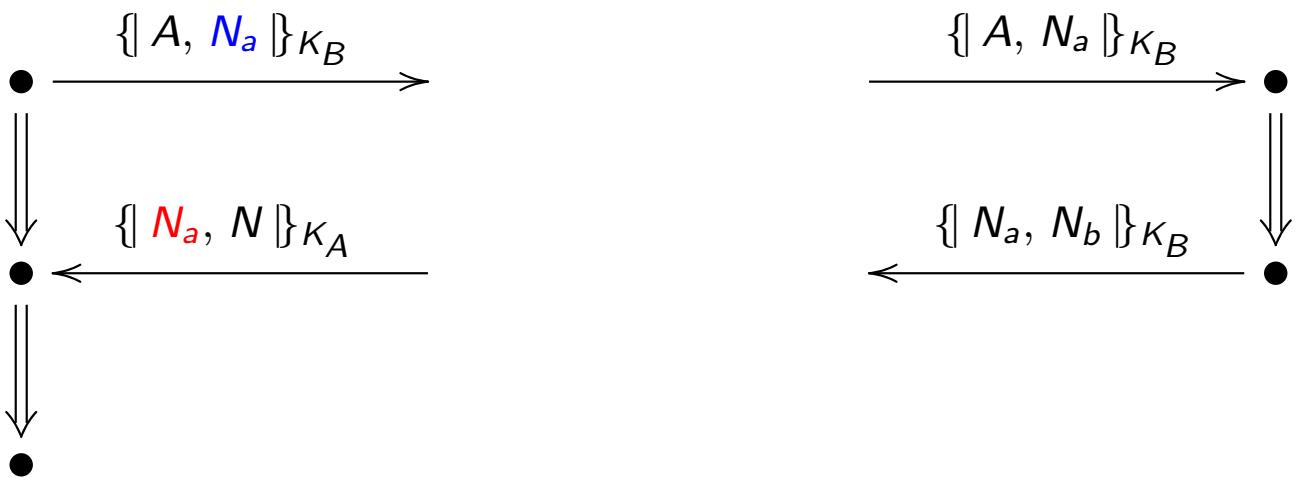
NS Analysis: Initiator's point of view

Assume K_A^{-1}, K_B^{-1} uncompromised, N_a fresh



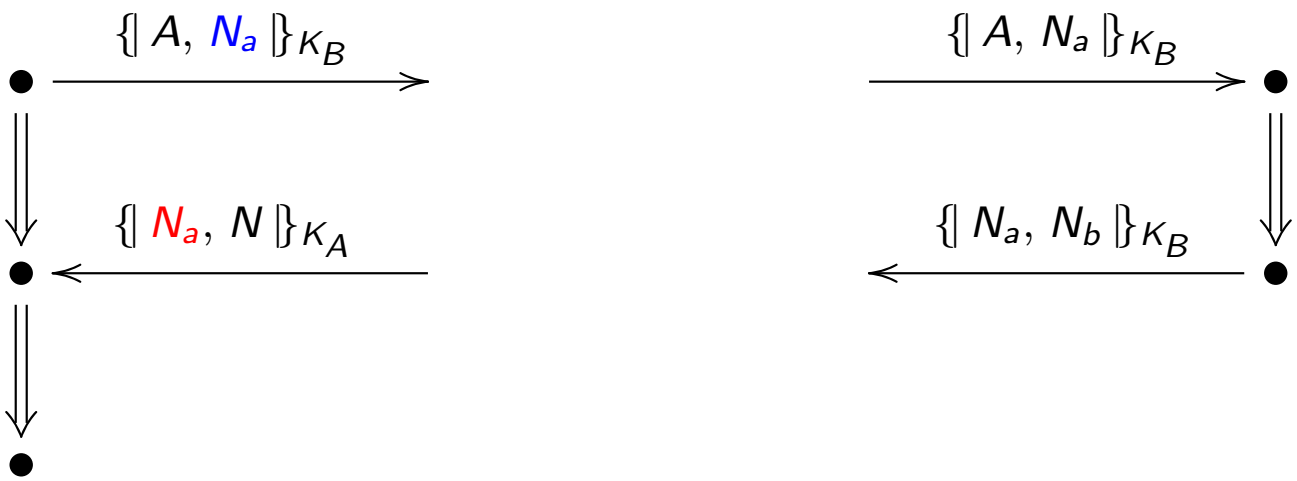
NS Analysis: Initiator's point of view

Assume K_A^{-1}, K_B^{-1} uncompromised, N_a fresh



NS Analysis: Initiator's point of view

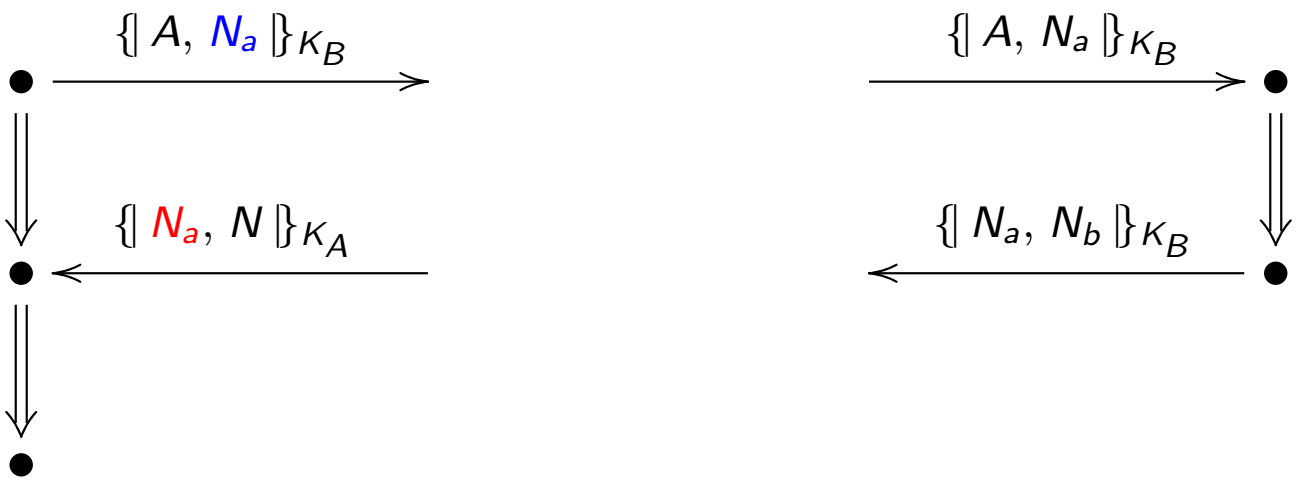
Assume K_A^{-1}, K_B^{-1} uncompromised, N_a fresh



Does $N = N_b$?

NS Analysis: Initiator's point of view

Assume K_A^{-1}, K_B^{-1} uncompromised, N_a fresh



Does $N = N_b$?

Yes, there are no available services!

Breaking and Proving

- How to break a protocol
  - Try to prove it correct
  - Where you get stuck, look for trouble
  - Specifically, look for unintended services to produce non-junk terms expected by regular principals

Breaking and Proving

- How to break a protocol
  - Try to prove it correct
  - Where you get stuck, look for trouble
  - Specifically, look for unintended services to produce non-junk terms expected by regular principals
- How to prove a protocol correct
  - Try to break it
  - See what unintended services must be used
  - "Read off" authentication properties

Breaking and Proving

- How to break a protocol
  - Try to prove it correct
  - Where you get stuck, look for trouble
  - Specifically, look for unintended services to produce non-junk terms expected by regular principals
- How to prove a protocol correct
  - Try to break it
  - See what unintended services must be used
  - "Read off" authentication properties
- Strand spaces: make these ideas precise, justify method

Breaking and Proving

- How to break a protocol
  - Try to prove it correct
  - Where you get stuck, look for trouble
  - Specifically, look for unintended services to produce non-junk terms expected by regular principals
- How to prove a protocol correct
  - Try to break it
  - See what unintended services must be used
  - "Read off" authentication properties
- Strand spaces: make these ideas precise, justify method

Thanks to collaborators:

Marco Carbone, Dan Dougherty,
Amy Herzog, Jonathan Herzog,
John Ramsdell, Javier Thayer, Lenore Zuck