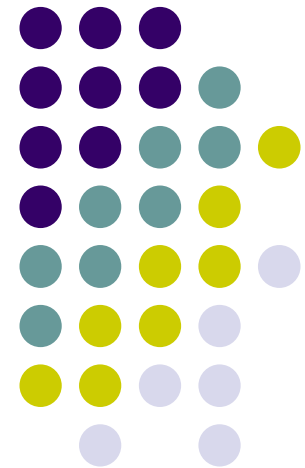


Extending the strand space model for security protocol composition

Genge Bela¹, Haller Piroska²

^{1,2} “Petru Maior” University of Targu Mures

Contact: {¹bgenge, ²phaller}@upm.ro





Introduction

- Security protocols?
 - Communication protocols + cryptography
 - Example message specification: $\{A, B, Na, K\}_{Kab}$
- Strands and strand spaces?
 - Strand: a sequence of send and receive events
 - Strand space: a collection of strands
 - Model protocol participants
 - Model intruder capabilities: Concat, Split, replay, Encrypt, Decrypt
- Composition?
 - Combining two or more protocols
 - Useful in design process
 - Verifying the independence of security protocols

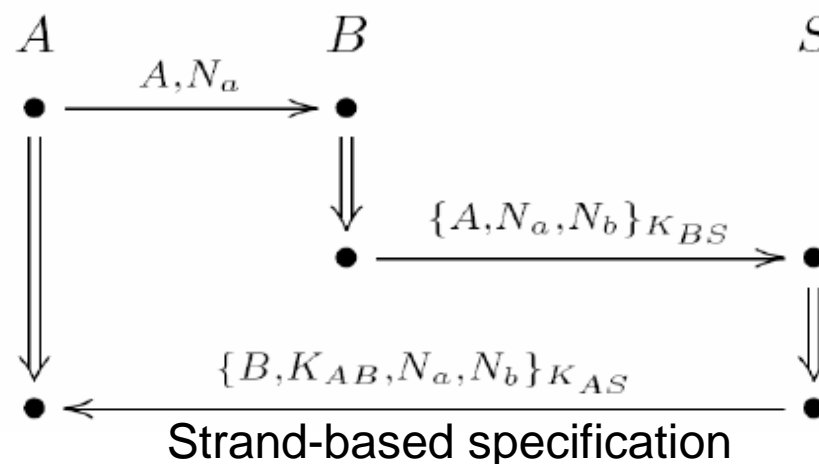


Why extending?

- Existing specification models are simplistic:
 - Informal purpose
 - Participant knowledge is not included in the message specification
 - Message components lack purpose information
 - The link between components is not made explicit
- Why extending the strand space model?
 - Highly general specification: allows the modeling of internal mechanisms: encryption, decryption, key generation
 - Flexibility

$A \rightarrow S: A, \{T_a, B, K_{ab}\}K_{as}$
 $S \rightarrow B: \{T_s, A, K_{ab}\}K_{bs}$
 $B \rightarrow A: \{N_b\}K_{ab}$
 $A \rightarrow B: \{N_{b+1}\}K_{ab}$

Regular specification



Compositionality requirements

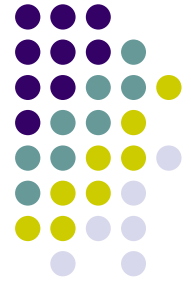
- Parallel composition -



- Protocol roles are merged together
- Purpose:
 - Create homogenous protocols with multiple security properties
 - Homogenous: initial protocol messages can not be clearly distinguished
- Requires:
 - Atomic message handling: respect message pre-conditioning
 - Verifying if the resulting messages respect the “strong independence” property
- Specification requirements:
 - Explicit message component-link specification
 - Explicit message sequence specification
 - Explicit modeling of participant knowledge

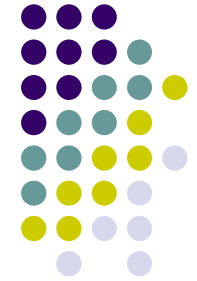
Compositionality requirements

- Sequential composition -



- Protocols are run sequentially, roles are not merged anymore
- Purpose:
 - Create protocols with multiple security properties
 - Messages belonging to the original protocols can be clearly distinguished
- Requires:
 - Verifying the independence of security protocols
- Specification requirements:
 - Explicit modeling of participant knowledge
 - For a syntactical analysis: model message patterns (verifiable message components)

Extending the regular strand model



- Explicit modeling of function names:

$FuncName ::=$	sk	(secret key)
	pk	(public key)
	pvk	(private key)
	h	(hash)

- Explicit term construction:

$$\mathcal{T} ::= . \mid R \mid N \mid K \mid (\mathcal{T}, \mathcal{T})$$

$$\mid \{\mathcal{T}\}_{FuncName(\mathcal{T})}$$

- The resulting strand model: $\langle \pm t_1, \pm t_2, \dots, \pm t_n \rangle \in (\pm \mathcal{T})^*$

$\left. \begin{array}{l} n_1 = \langle s, i \rangle \\ n_2 = \langle s, i+1 \rangle \\ strand(n_1) = strand(n_2) \end{array} \right\}$	$n_1 \Rightarrow n_2$	$\left. \begin{array}{l} n_1, n_2 \in \mathcal{N} \\ strand(n_1) \neq strand(n_2) \end{array} \right\}$	$n_1 \rightarrow n_2$
--	-----------------------	---	-----------------------

Adding binding terms for parallel composition



- Message types (i.e. patterns):

$$\begin{aligned} \text{BasicTT} ::= & r && (\text{role type}) \\ & | n && (\text{nonce type}) \\ & | k && (\text{key type}) \\ & | b && (\text{binding type}) \end{aligned}$$

- Basic term: $\langle \rho, \nu, \kappa, \beta, f, k, \theta \rangle$

$$\rho \in R^* \quad \nu \in N^* \quad \kappa \in K^* \quad \beta \in B^* \quad f \in \text{FuncName} \quad k \in K \quad \theta \in \text{BasicTT}^*$$

- Function mappings and classifiers:

$$\text{Roles}(b) = \rho, \text{Nonces}(b) = \nu, \text{Keys}(b) = \kappa, \quad C ::= C_{\mathcal{R}} \quad (\text{Role classifier})$$
$$\text{Bindings}(b) = \beta, \text{Func}(b) = f, \text{BindingKey}(b) = k, \quad | C_{\mathcal{K}} \quad (\text{Knowledge classifier})$$
$$\text{TypeSet}(b) = \theta$$

- Introducing b-strands: $s : (\pm B)^* \times C$

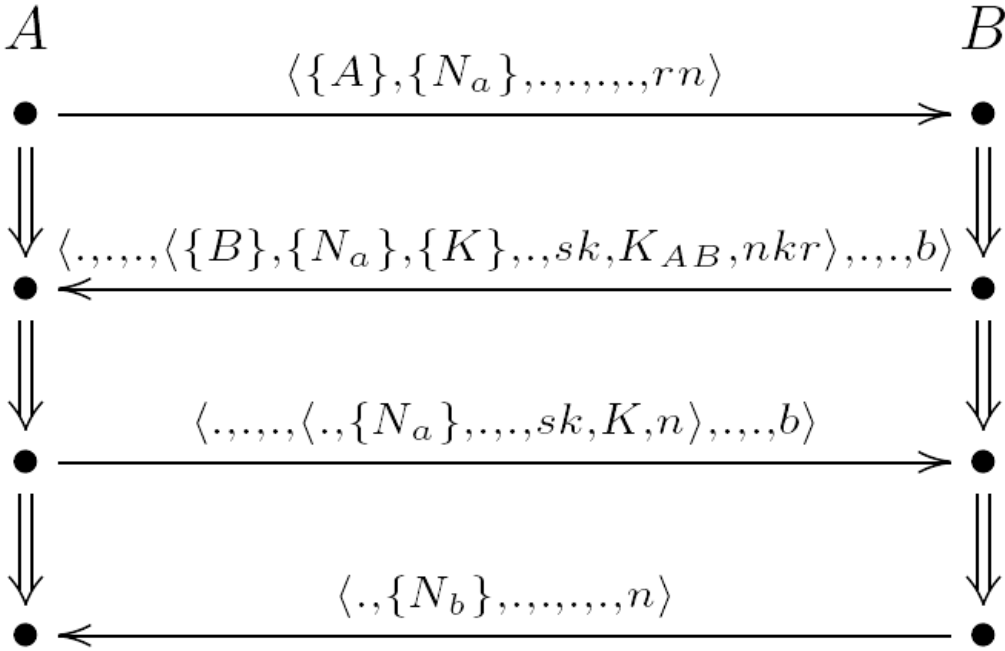
B-strand specification example



- Lowe’s modified “BAN concrete Andrew Secure RPC”

$A \rightarrow B: A, N_a$
 $B \rightarrow A: \{N_a, K, B\}_{K_{AB}}$
 $A \rightarrow B: \{N_a\}_K$
 $B \rightarrow A: N_b$

Regular specification

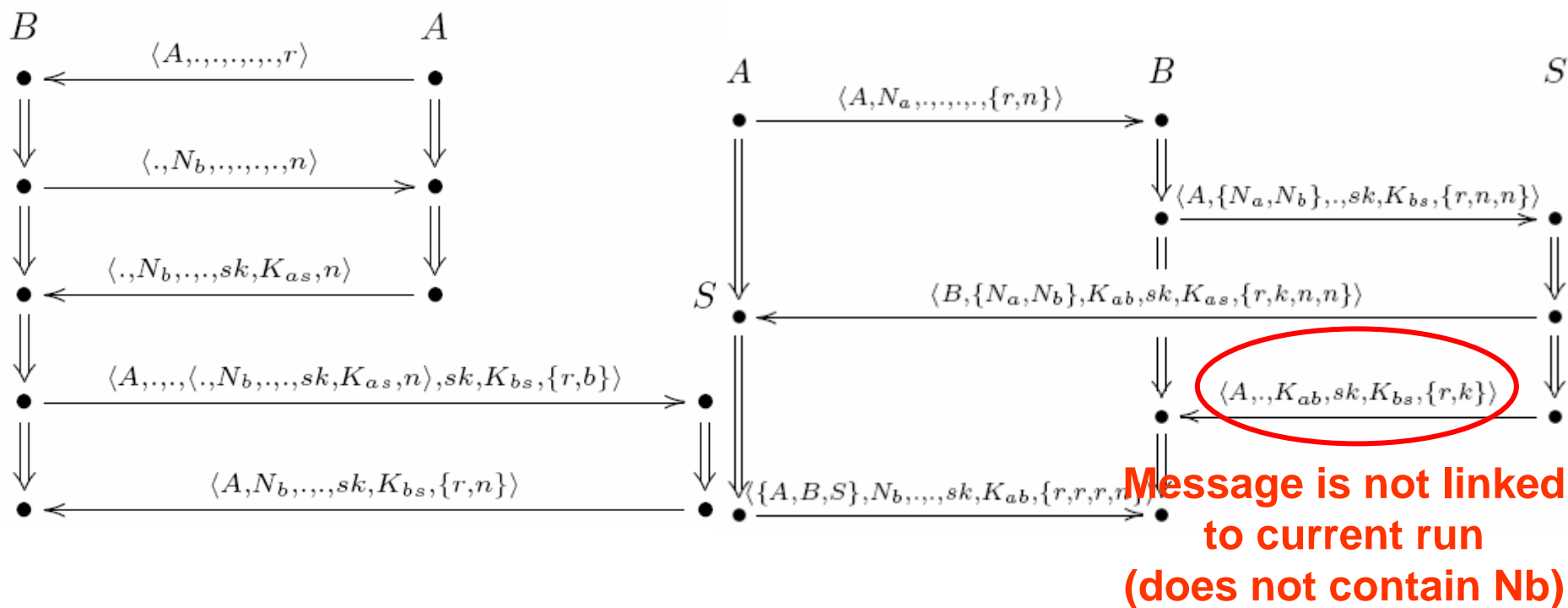


B-strand-based specification

Parallel composition example



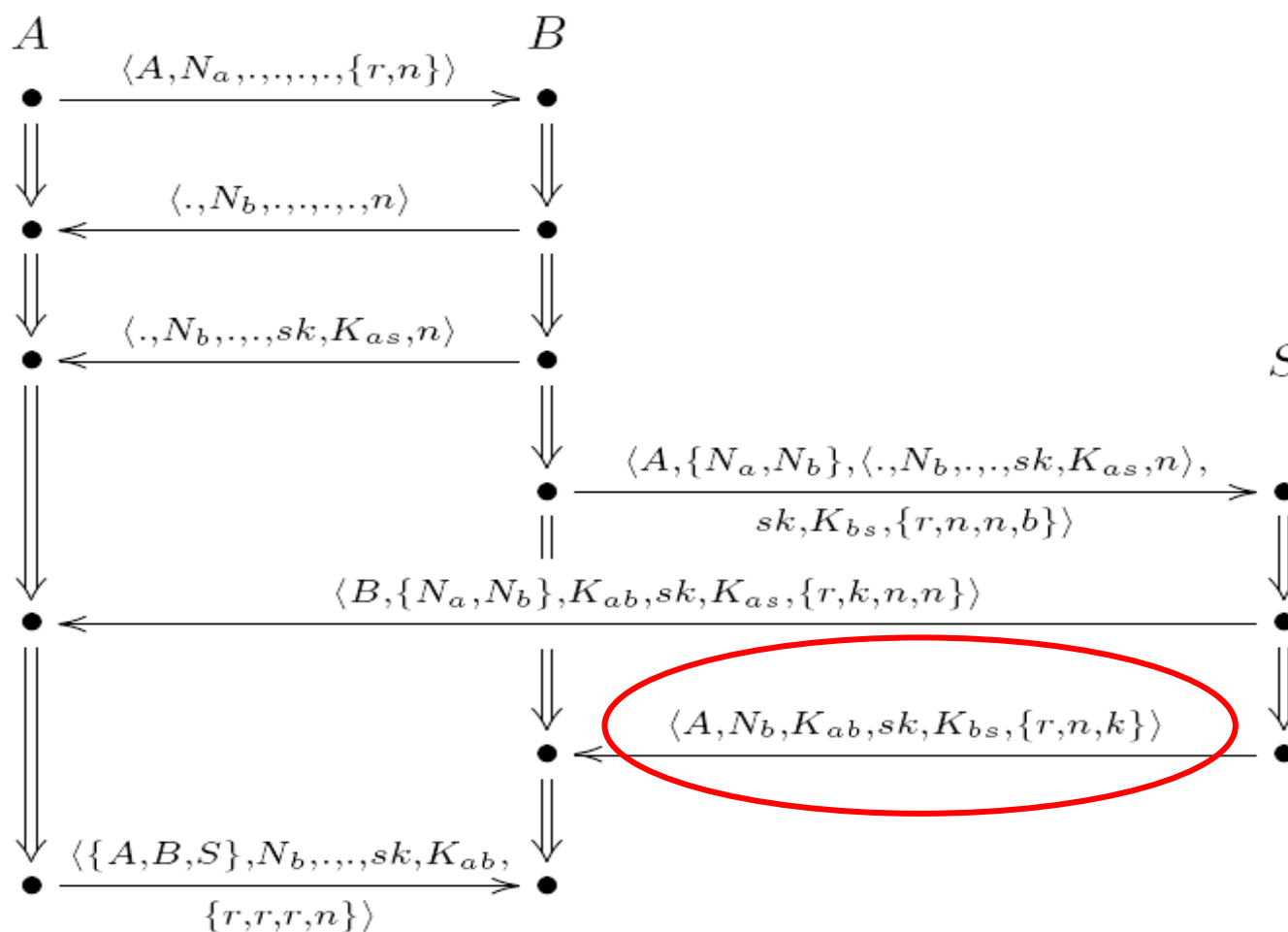
- Woo and Lam Pi 3
 - One-way authentication
- Lowe's modified Yahalom
 - Two way authentication
 - Key exchange
 - One of the authentication messages is flawed





Resulting protocol

- Solves the authentication problem



Adding typed knowledge terms for sequential composition



- Extending basic types:

$$\begin{array}{l} \text{BasicTT} ::= \text{BasicTT} \\ | \text{K}_t \quad (\text{typed keys}) \\ | \text{u} \quad (\text{unknown type}) \end{array}$$
- Defining terms: $\mathcal{T}_t ::= . | \text{BasicTT} | (\mathcal{T}_t, \mathcal{T}_t)$

$$| \{\mathcal{T}_t\}_{\text{FuncName}(\mathcal{T}_t)}$$
- Re-defining strand spaces (t-strands):

$$\langle \pm t_1, \pm t_2, \dots, \pm t_n \rangle \in (\pm \mathcal{T}_t)^*$$
- Mapping functions:

$$TTr(t) = \begin{cases} r, & \text{if } t \in \mathbb{R}, \\ n, & \text{if } t \in \mathbb{N}, \\ k_i, & \text{if } t \in \text{K}, \\ (TTr(t_1), TTr(t_2)), & \text{if } t = (t_1, t_2), \\ \{TTr(t_1)\}_{f(TTr(t_2))}, & \text{if } t = \{t_1\}_f(t_2). \end{cases}$$

$$KTT_r(\kappa, t) = \begin{cases} (KTT_r(\kappa, t_1), KTT_r(\kappa, t_2)), & \text{if } t = (t_1, t_2), \\ \{KTT_r(\kappa, t_1)\}_{f(KTT_r(\kappa, t_2))}, & \text{if } t = \{t_1\}_f(t_2), \\ TTr(t), & \text{if } t \in \kappa, \\ u, & \text{otherwise.} \end{cases}$$

T-Strand specification example

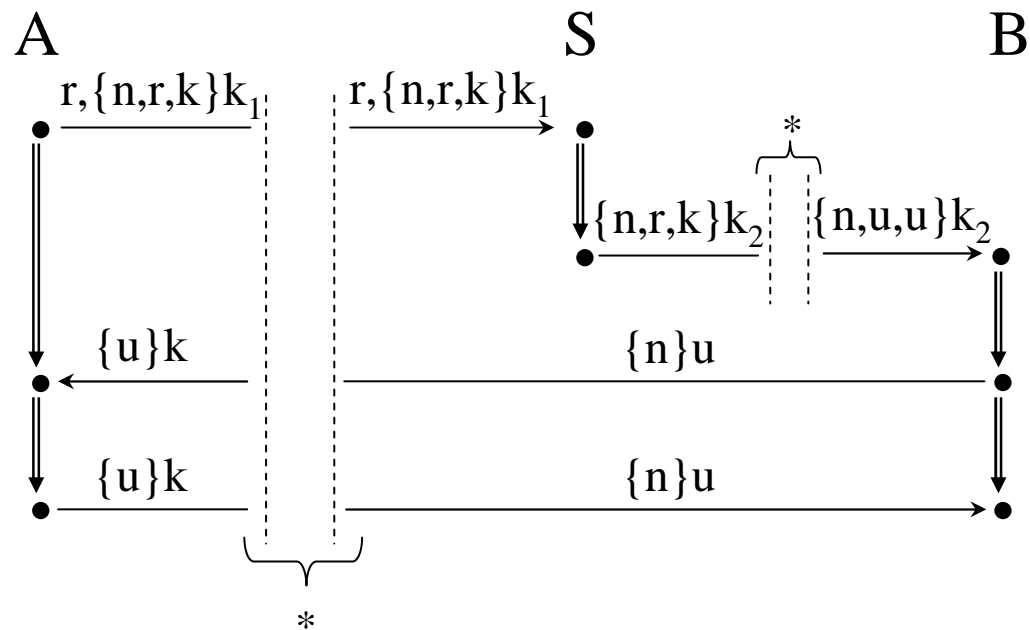


- Lowe's modified Wide Mouthed Frog:

$A \rightarrow S: A, \{Ta, B, Kab\}Kas$
 $S \rightarrow B: \{Ts, A, Kab\}Kbs$
 $B \rightarrow A: \{Nb\}Kab$
 $A \rightarrow B: \{Nb+1\}Kab$

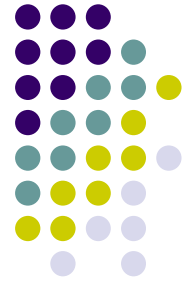
where

A, B, S – participant names
 Ta, Ts – timestamps
 Kas, Kbs – long-term keys
 Kab – session key
 Nb – nonce



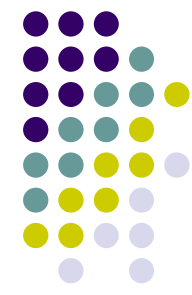
* Transformation layers based on role knowledge

Sequential composition example



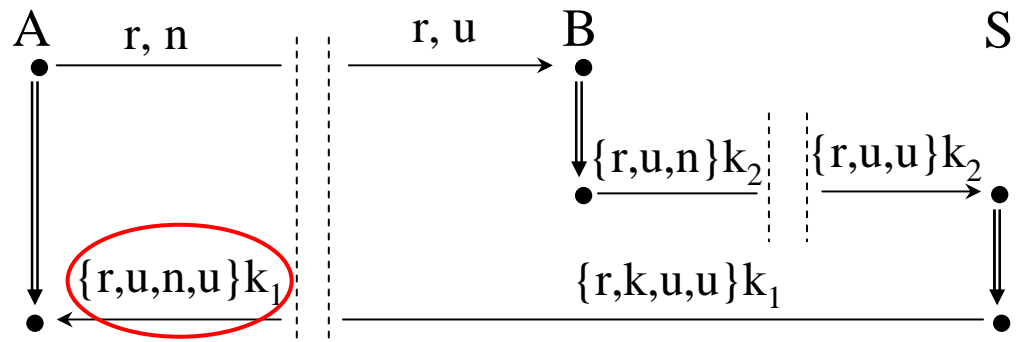
- Implies verifying the independence:
 - Existing security properties remain valid after composition
- Verification principle:
 - Analyze intruder's powers: Encrypt, Decrypt, Concat, Split, Replay
 - Analyze message structures in the context of the intruder

Sequential composition example

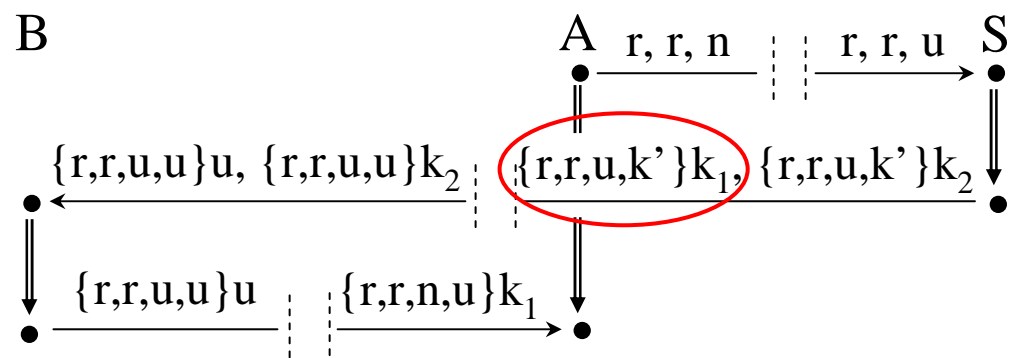


- Key exchange parts from the Yahalom-Lowe (Y-L) and Kao-Chow (K-C) protocols

(Y-L)
 $A \rightarrow B: A, Na$
 $B \rightarrow S: \{A, Na, Nb\}K_{bs}$
 $S \rightarrow A: \{B, Kab, Na, Nb\}K_{as}$



(K-C)
 $A \rightarrow S: A, B, N'a$
 $S \rightarrow B: \{A, B, N'a, K'ab\}K_{as},$
 $\{A, B, N'a, K'ab\}K_{bs}$
 $B \rightarrow A: \{A, B, N'a, K'ab\}K_{as}$





Correcting the problem

- Adding more verifiable information to the Y-L message:
 - $\{B, K_{ab}, N_a, N_b\}_{K_{as}} \rightarrow \{A, B, K_{ab}, N_a, N_b\}_{K_{as}}$
- The problem:
 - Other protocols can still be used to construct attacks
- More complex solutions:
 - Using tagging schemes to prevent type-flaw attacks:
 - $\{B, K_{ab}, N_a, N_b\}_{K_{as}} \rightarrow \{B, K_{ab}, N_a, N_b, (n, k, n, n)\}_{K_{as}}$
 - Adding protocol information to prevent replay attacks:
 - $\{B, K_{ab}, N_a, N_b\}_{K_{as}} \rightarrow B, K_{ab}, N_a, N_b, (P1, V, M_x, n, k, n, n)\}_{K_{as}}$



Future work

- Develop a unifying Meta-model
- Add performance-related information to optimize the performance of the created protocols
- Implement a tool for the automated composition of security protocols

Thanks for your attention!



- Questions?