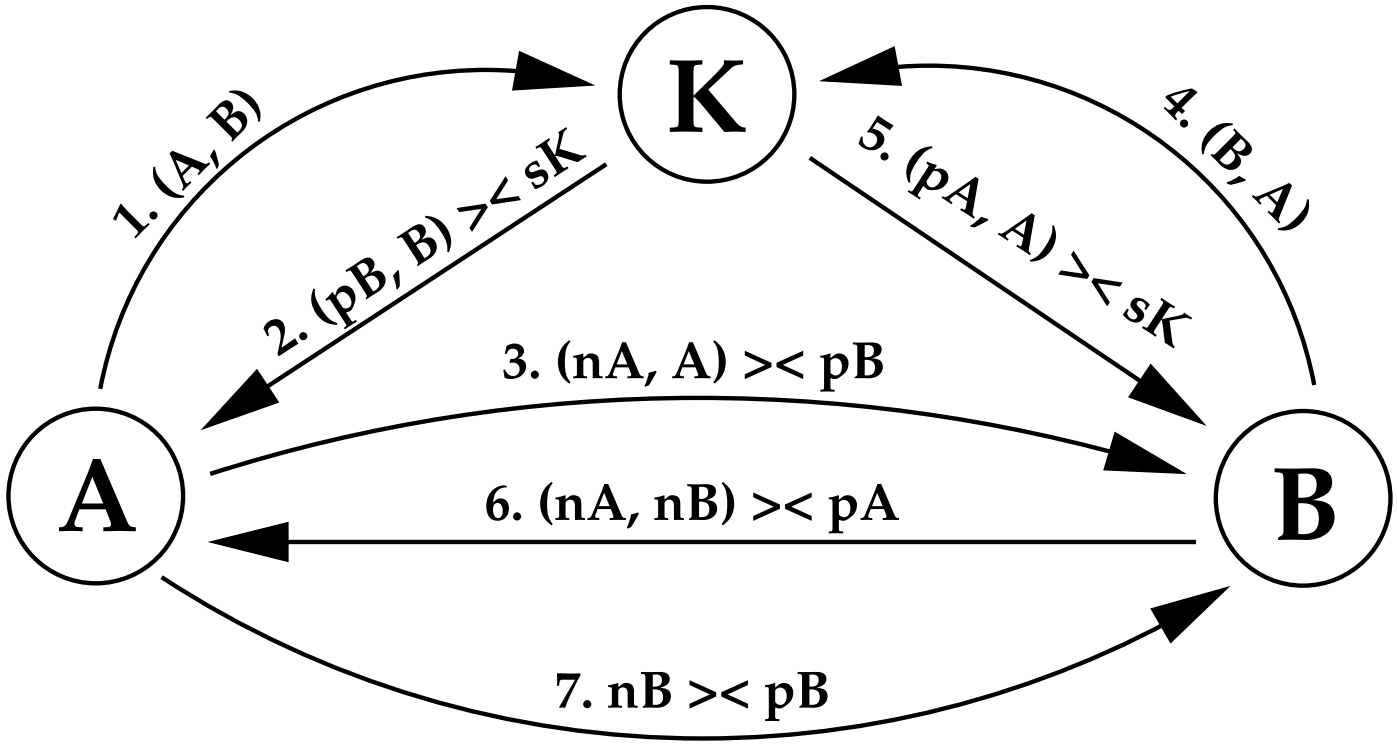
Kryptographische Protokolle und Model Checking

Martin Horsch

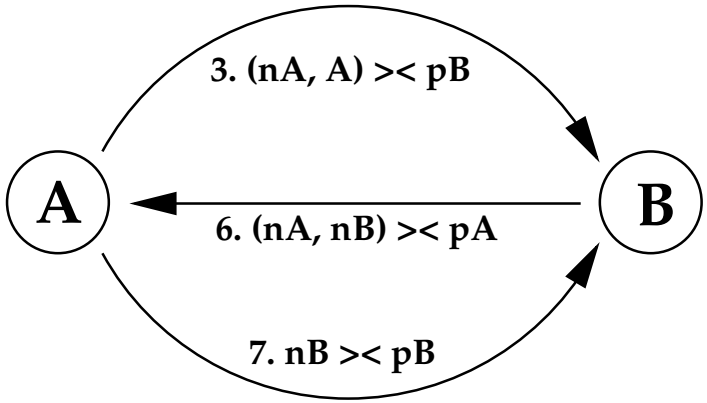
5. Juli 2004

Needham-Schroeder-Protokoll

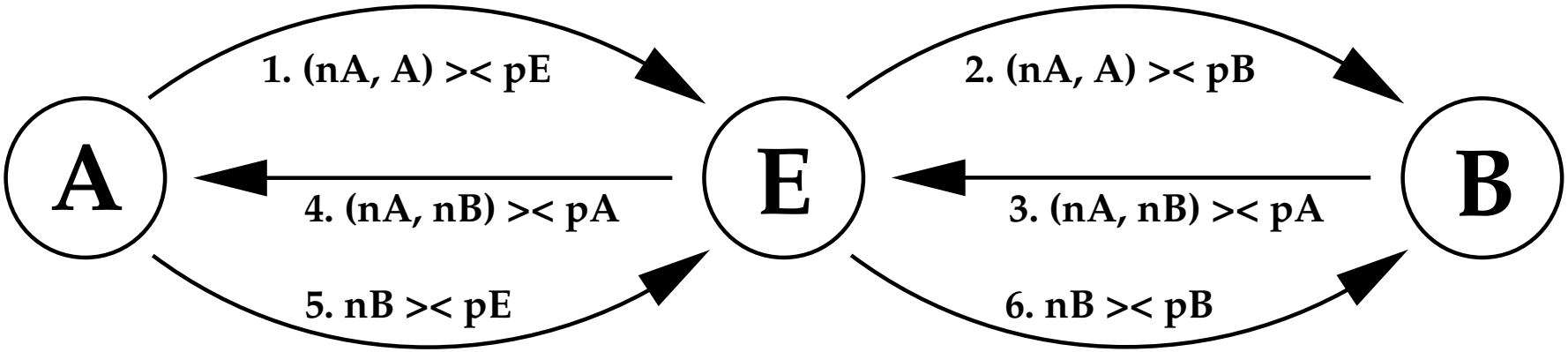


Wenn wir annehmen, dass allen Teilnehmern die public keys aller anderen Teilnehmer schon bekannt sind, können wir den Keyserver und die Schritte **1**, **2**, **4** und **5** des Protokolls weglassen.

reduziertes Needham-Schroeder-Protokoll

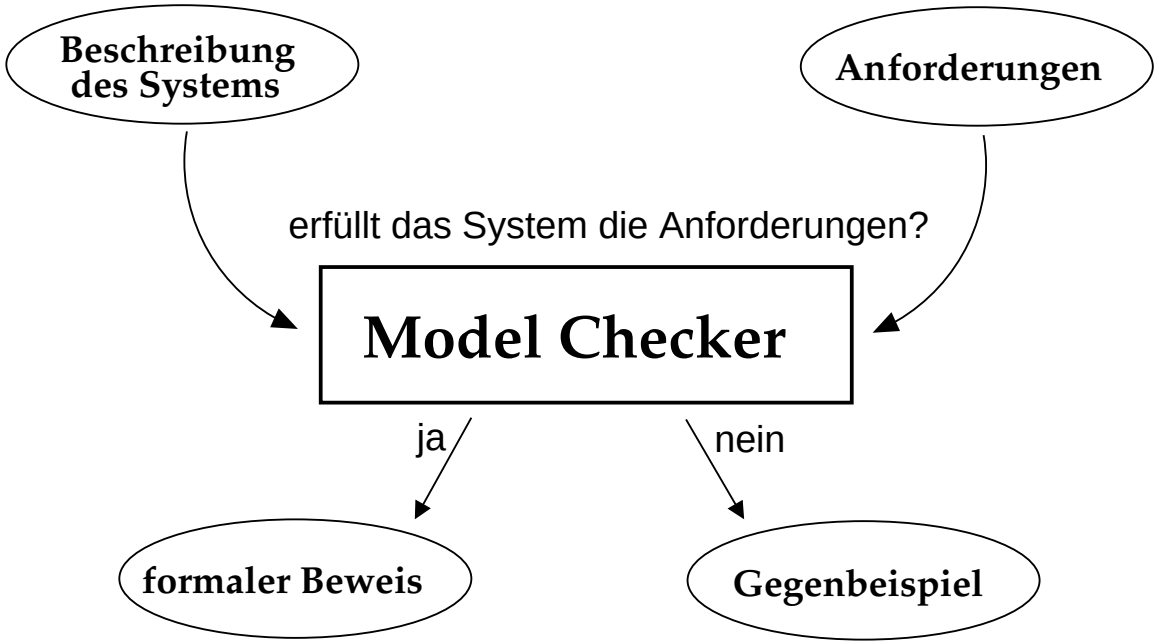


Nach **Gavin Lowe** ist dieses Protokoll angreifbar:



formale Verifikation eines Protokolls

Wir wollen die Beschreibung des Protokolls und unsere Erwartungen daran so formalisieren, dass seine Korrektheit mit geeigneter Software bewiesen werden kann.



Ansatz dafür: Untersuchung des gesamten Zustandsraumes

System und Anforderungen

Prozesstypen repräsentieren im Protokoll auftretende Rollen:

Prozesstyp α (<Name> A, <Partner> Y):
1. erzeuge eine unbestätigte Session i von A mit Y
2. sende (<Nonce> nA, A) $><$ $pY \longrightarrow Y$
3. warte auf (nA, v) $><$ pA und merke v (Nonce von Y)
4. bestätige die Session i
5. sende v $><$ $pY \longrightarrow Y$

Prozesstyp β (<Name> B):
1. warte auf (w, Z) $><$ pB und merke w und Z (Nonce und Name des Partners)
2. erzeuge eine unbestätigte Session j von B mit Z
3. sende (w, <Nonce> nB) $><$ $pZ \longrightarrow Z$
4. warte auf nB $><$ pB
5. bestätige die Session j

Für alle Teilnehmer P_1, P_2 soll gelten:
gibt es eine bestätigte Session von P_1 mit P_2, dann gibt es auch mindestens eine Session von P_2 mit P_1.

Untersuchung des Zustandsraumes

Im Startzustand unseres Systems existieren keine Sessions zwischen Teilnehmern. Wir wollen alle von da aus erreichbaren Zustände finden, in denen die Anforderung an das Protokoll verletzt ist. Deshalb beschreiben wir den Eindringling E als so mächtig wie es innerhalb unseres Modells denkbar ist *(Dolev-Yao-Modell)*:

1. E fängt alle Kommunikation zwischen Teilnehmern ab.
2. E merkt sich alle Daten aus abgefangenen Nachrichten.
3. E **kann beliebig** Nachrichten daraus konstruieren.
4. E **kann beliebig** neue Daten anlegen.
5. E **kann** als normaler Teilnehmer in Erscheinung treten.

E geht nicht systematisch, sondern beliebig vor, deshalb ist ein **nichtdeterministisches** Modell besonders geeignet, um ihn zu beschreiben. Da E nicht durch aufwendige Heuristiken simuliert wird, kann man seine Prozessbeschreibung **automatisch** generieren.

Begrenzung des Zustandsraumes

Der Zustandsraum dieses Systems ist **unendlich**, denn jede Nonce kann beliebige Werte annehmen und der Eindringling kann beliebig komplexe Nachrichten aufbauen.

Deshalb ist es notwendig, ein System mit endlicher Zustandsmenge zu konstruieren, das entweder äquivalent ist oder unwesentlich mehr Angriffe findet als tatsächlich möglich sind.

Freie Variablen wie Noncen oder von E erzeugte Werte werden dabei als paarweise voneinander verschiedene **Symbole** behandelt. Beim Needham-Schroeder-Protokoll sind das $\mathfrak{n}A$, $\mathfrak{n}B$ und ein generisches Datum $\mathfrak{g}D$, das von E beliebig verwendet werden kann.

Welche Daten kann der Eindringling sammeln?

Beim reduzierten NSP mit zwei Teilnehmern (und E) sieht man:

1. die Menge der Teilnehmer ist $\mathfrak{U} = \{\mathrm{A}, \mathrm{B}, \mathrm{E}\}$ und die Menge der Variable $\mathcal{V} = \{\mathfrak{n}A, \mathfrak{n}\mathrm{B}, \mathfrak{g}\mathrm{D}\}$.

2. das Anfangswissen von E ist $\{\mathrm{A}, \mathrm{B}, \mathrm{E}, \mathfrak{g}\mathrm{D}\}$.

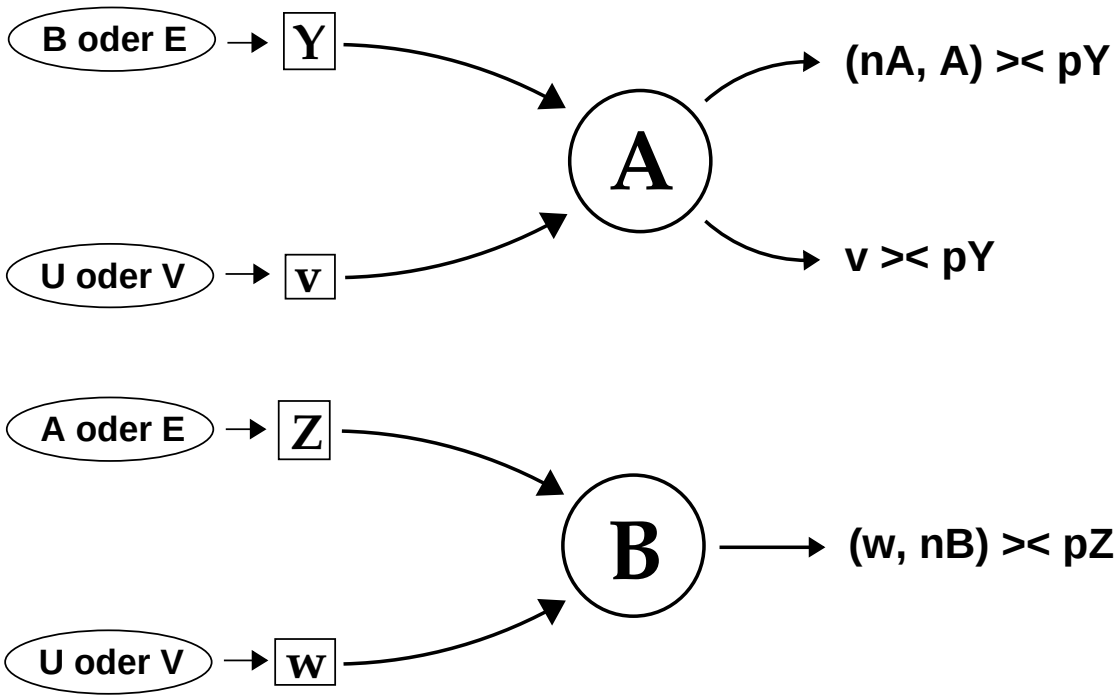
3. alle Nachrichten bestehen aus Elementen von $\mathfrak{U} \cup \mathcal{V}$.

4. alle Nachrichten, die von einem Teilnehmer erstellt werden können, haben entweder das Format $(\mathfrak{d}, e) >< \mathfrak{p}W$ oder $\mathfrak{d} >< \mathfrak{p}W$ mit $\mathfrak{d}, e \in (\mathfrak{U} \cup \mathcal{V}), W \in \mathfrak{U}$.

E kann sich nur Bestandteile solcher Nachrichten merken, die einer der anderen Teilnehmer erstellen kann.

Datenflussanalyse

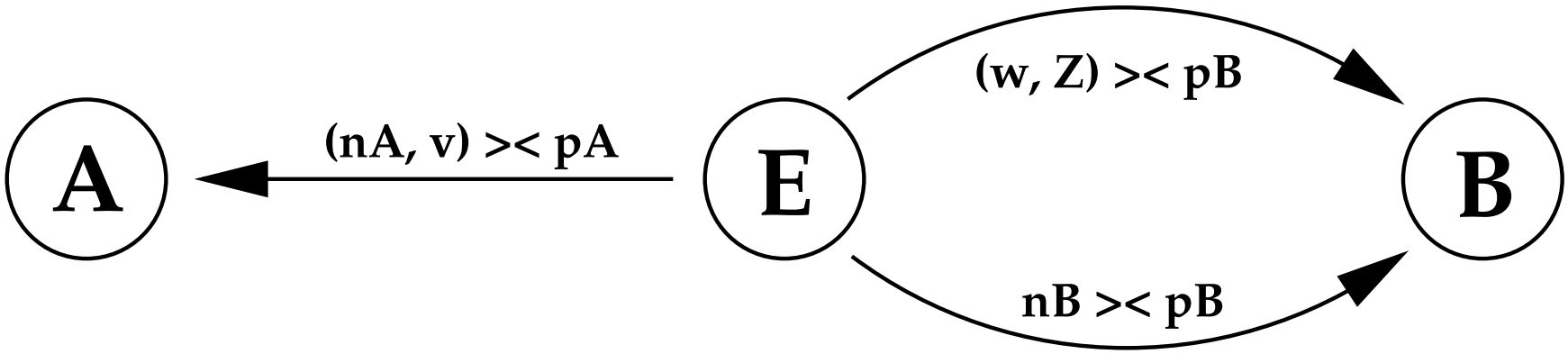
Durch eine Analyse der beiden Prozesstypen α und β kann man die Menge der potentiell abgefangenen Nachrichten weiter eingrenzen:



Hier ist A die Instanz von α und B die Instanz von β.
E kann $2 + 6 \cdot 2 + 6 \cdot 2 = \mathbf{26}$ verschiedene Nachrichten abfangen.

Welche Nachrichten kann E senden?

Aus dem Protokoll ergibt sich, welche Nachrichten von A bzw. B akzeptiert werden. Nur solche Nachrichten sollte der modellierte Eindringling auch erzeugen.



Wie vorhin sind $v, w \in U \cup V$ und $Z \in \{A, E\}$.
E kann also nur $6 + 6 \cdot 2 + 1 = \mathbf{19}$ verschiedene Nachrichten senden.

automatische Generierung eines Prozesstyps für E

Komponenten einer Nachricht sind hier ihre von E lesbaren Bestandteile. Die Mengen der Teilnehmer U und der Nachrichten- und Komponentenformate F seien bekannt. Eine einzelne Variable habe das Format $\mathrm{f_V} \in \mathrm{F}$.

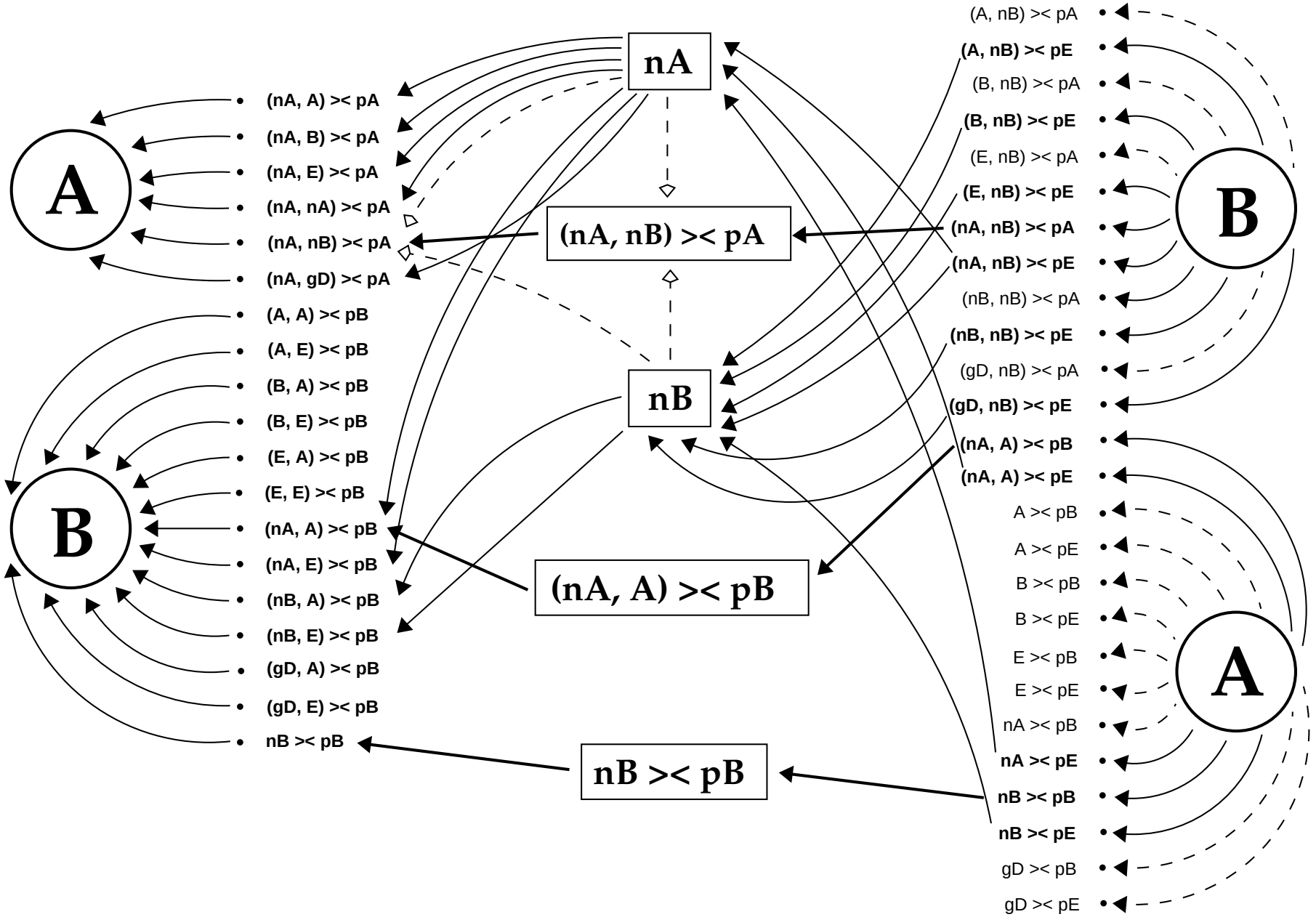
1. bestimme die symbolischen Variablen und das Anfangswissen $\mathrm{K_0} \subseteq \mathrm{V} \times \{\mathrm{f_V}\}$.

2. bestimme die Menge aller von Teilnehmern *akzeptierten* Nachrichten und ihrer Komponenten $\mathfrak{msg}_\mathrm{E} \subseteq \mathfrak{comp}_\mathrm{E} \subset (\mathrm{U} \cup \mathrm{V})^* \times \mathrm{F}$.

3. bestimme die Menge aller von Teilnehmern *generierbaren* Nachrichten und ihrer Komponenten $\mathfrak{msg}_\mathrm{U} \subseteq \mathfrak{comp}_\mathrm{U} \subset (\mathrm{U} \cup \mathrm{V})^* \times \mathrm{F}$.

4. $\tilde{\mathrm{K}} := (\mathfrak{comp}_\mathrm{E} \cap \mathfrak{comp}_\mathrm{U}) \setminus \mathrm{K_0}$ enthält alle für E brauchbaren Komponenten. E soll sich genau die Komponenten aus $\mathrm{K} := \tilde{\mathrm{K}} \setminus \tilde{\mathrm{K}}^+$ merken können.

Der Wissenszustand von E kann dann in $|\,\mathrm{K}\,|$ Flags gespeichert werden. Beim NSP ergibt sich $\mathrm{K} = \{\mathfrak{n}\mathrm{A},\ \mathfrak{n}\mathrm{B},\ (\mathfrak{n}\mathrm{A}, \mathrm{A}) >\!< \mathfrak{p}\mathrm{B},\ (\mathfrak{n}\mathrm{A}, \mathfrak{n}\mathrm{B}) >\!< \mathfrak{p}\mathrm{A},\ \mathfrak{n}\mathrm{B} >\!< \mathfrak{p}\mathrm{B}\}$. Bei geeigneter Implementierung kann man dann alle Nachrichten ignorieren, aus denen E in keinem Fall etwas lernt. (Beim NSP: **13** statt 26.)


nA
(nA, nB) >< pA
nB
(nA, A) >< pB
nB >< pB
A
B
(nA, A) >< pA
(nA, B) >< pA
(nA, E) >< pA
(nA, nA) >< pA
(nA, nB) >< pA
(nA, gD) >< pA
(A, A) >< pB
(A, E) >< pB
(B, A) >< pB
(B, E) >< pB
(E, A) >< pB
(E, E) >< pB
(nA, A) >< pB
(nA, E) >< pB
(nB, A) >< pB
(nB, E) >< pB
(gD, A) >< pB
(gD, E) >< pB
nB >< pB
B
A
(A, nB) >< pA
(A, nB) >< pE
(B, nB) >< pA
(B, nB) >< pE
(E, nB) >< pA
(E, nB) >< pE
(nA, nB) >< pA
(nA, nB) >< pE
(nB, nB) >< pA
(nB, nB) >< pE
(gD, nB) >< pA
(gD, nB) >< pE
(nA, A) >< pB
(nA, A) >< pE
A >< pB
A >< pE
B >< pB
B >< pE
E >< pB
E >< pE
nA >< pB
nA >< pE
nB >< pB
nB >< pE
gD >< pB
gD >< pE

Implementierung mit SPIN

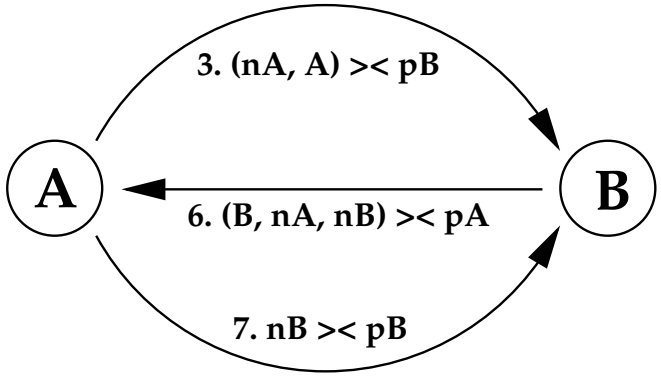
Der Model Checker *SPIN* wurde von Gerard Holzmann bei den Bell Labs entwickelt. Erste Version 1981 unter dem Namen *pan* (protocol analyzer) auf Grundlage von Prozessalgebren, ab 1983 *Trace* (mit Büchi-Automaten). 1989 Einführung der folgenden Struktur:

Einschränkungen für das Modell:
Systembeschreibung in *PROMELA* (process meta-language)
Formulierung der Anforderungen:
z.B. in *LTL* (linear temporal logic) oder als *Büchi-Automat*

SPIN steht für *simple PROMELA interpreter*.

Mehr Infos zu SPIN: http://spinroot.com/

Lowes Korrekturvorschlag



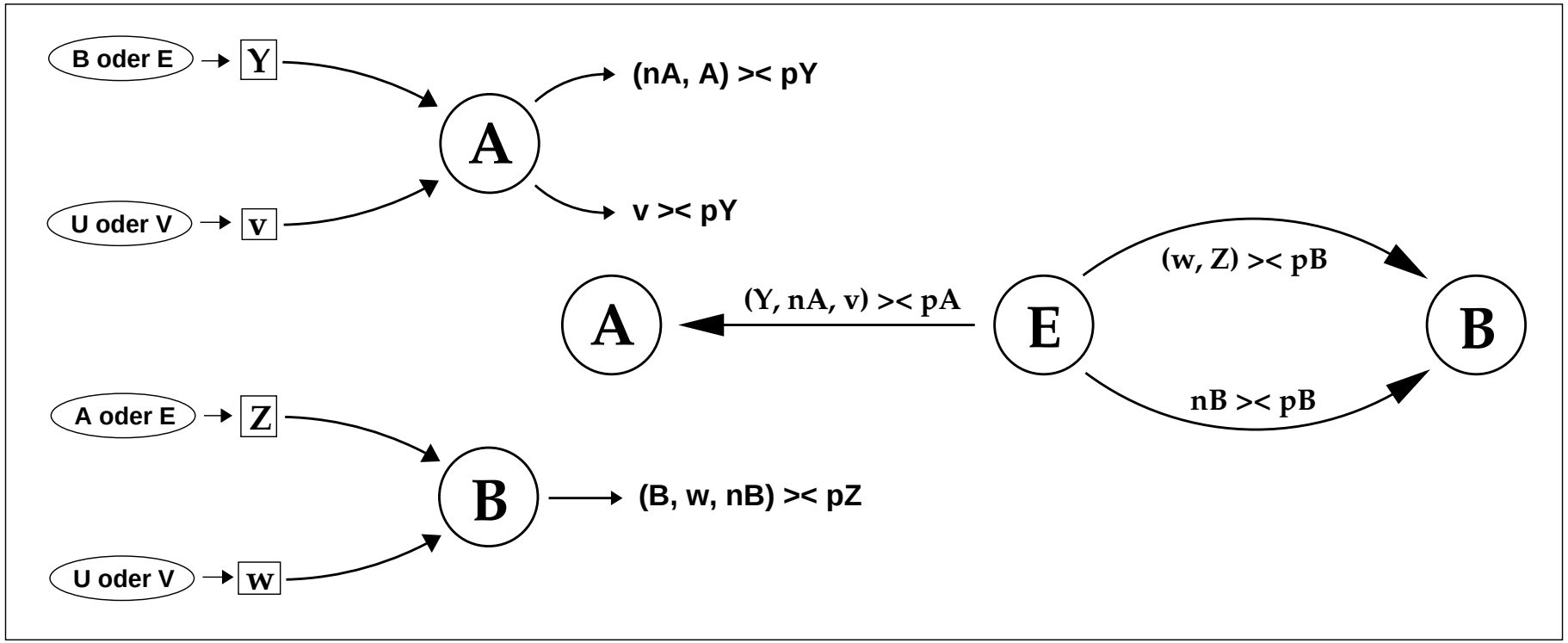
Prozesstyp α (<Name> A, <Partner> Y):

1. erzeuge eine unbestätigte Session i von A mit Y
2. sende (<Nonce> nA, A) >< pY $\longrightarrow$ Y
3. warte auf (**Y**, nA, v) >< pA und merke v (Nonce von Y)
4. bestätige die Session i
5. sende v >< pY $\longrightarrow$ Y

Prozesstyp β (<Name> B):

1. warte auf (w, Z) >< pB und merke w und Z (Nonce und Name des Partners)
2. erzeuge eine unbestätigte Session j von B mit Z
3. sende (**B**, w, <Nonce> nB) >< pZ $\longrightarrow$ Z
4. warte auf nB >< pB
5. bestätige die Session j

Analyse des verbesserten Protokolls



Dann ist $K = \{nA,\ nB,\ (nA, A) >< pB,\ (B, nA, nB) >< pA,\ nB >< pB\}$.

einige weitere Ansätze

FDR: von **Lowe** 1996 auf NSP angewendeter CSP-Model-Checker

NRL: Erzeugung von Invarianten (Meadows - 1996)

Darstellung krypt. Protokolle als Petri-Netze (Crazzolara, Winskel)

Analyse der Kombination aus Protokoll und Verschlüsselung durch Umformung in quadratische Gleichungssysteme (Millen, Shmatikov)

Ausblick

BAN-Logik: notiert Glaubenszustände (Burrows, Abadi, Needham - 1990)

Spi-Kalkül: Prozesskalkül (Abadi, Gordon - 1997); Grundlage: π-Kalkül