Protokollsicherheit

Informationsfluss

Informationsfluss

Bisherige Analysen:

- Geheimhaltung (Secrecy): Verifiziere, dass Geheimnisse nicht über einen öffentlichen Kanal geschickt und so dem Angreifer bekannt werden.
- Korrespondenzeigenschaften

Der Angreifer könnte auch Informationen über Geheimnisse erhalten, ohne dass diese komplett veröffentlicht werden.

Beispiel:

$$\nu k.\, \text{let } m_1 = \text{encrypt}(x, k) \text{ in let } m_2 = \text{encrypt}(y, k) \text{ in } \overline{c}\langle m_1.m_2 \rangle$$

Die Geheimnisse x und y selbst werden nicht veröffentlicht. Der Angreifer weiß jedoch, ob sie gleich sind.

Informationsfluss

Information über ein Geheimnis kann auch indirekt bekannt werden.

Beispiel: In einem Protokoll zur Authentifizierung einer Banküberweisung wird eine zusätzliche Nachricht mit einer Erinnerung verschickt, dass der aktuelle TAN-Block bald aufgebraucht ist.

Selbst wenn der TAN-Block selbst dann mit der Post verschickt wird, wird somit Information über die Anzahl der bereits verbrauchten TANs bekannt.

$$\text{if } f(\text{secret}) = M \text{ then } P \text{ else } Q$$

Nicht-Interferenz

Dass keine Information über ein Geheimnis bekannt wird, kann für den pi Kalkül mittels *Observationsäquivalenz* formuliert werden.

Informell: Ein Angreifer kann keinen Unterschied feststellen, wenn die Geheimnisse durch andere Werte ersetzt werden.

Strong Secrecy: Sei P ein Prozess, mit freien Variablen x_1,...,x_n. Der P hält die Variablen x_1,...,x_n geheim falls

$$P\{M_1/x_1, \ldots, M_n/x_n\} \approx P\{N_1/x_1, \ldots, N_n/x_n\}$$

für alle geschlossenen Terme $M_1, \ldots, M_n$ und $N_1, \ldots, N_n$ gilt.

Observationsäquivalenz

Schreibe wieder $\Longrightarrow$ für $\longrightarrow^*$.

Schreibe $P \Downarrow a$ falls $P \Longrightarrow E[\overline{a}\langle M\rangle.P]$ gilt für geeignete P und M sowie einen Evaluationskontext $E[-]$, der a nicht bindet.
(P kann nach ≥ 0 internen Schritten auf Kanal a senden.)

Definition:
Observationsäquivalenz $\approx$ ist die größte symmetrische Relation, mit folgenden Eigenschaften (für alle P, Q, P'):

1. Aus $P \approx Q$ und $P \Downarrow a$ folgt $Q \Downarrow a$ für alle a.
2. Gilt $P \approx Q$ und $P \Longrightarrow P'$, so gibt es Q' mit $Q \Longrightarrow Q'$ und $P' \approx Q'$.
3. Aus $P \approx Q$ folgt $E[P] \cong E[Q]$ für jeden Auswertungskontext $E[-]$.

Nicht-Interferenz

Beispiel:

$$P(x,y) = \nu k. \quad \begin{array}{l} \text{let } m_1 = \text{encrypt}(x,k) \text{ in} \\ \text{let } m_2 = \text{encrypt}(y,k) \text{ in} \\ \overline{c}\langle m_1.m_2 \rangle \end{array}$$

Für $M \neq N$ gilt nicht $P(M,M) \not\approx P(M,N)$:
Andernfalls müsste nach Bed. 3 mit

$$E[-] \;:=\; [-] \;\mid\; c(z).\,\text{if } \text{fst}(z) = \text{snd}(z) \text{ then } \nu k.\,\overline{d}\langle k \rangle \text{ else } 0$$

auch $E[P(M,M)] \approx E[P(M,N)]$ gelten.
Das kann aber nicht sein, da Bed. 1 nicht gilt: $E[P(M,M)] \Downarrow d$ und $E[P(M,N)] \not\Downarrow d$.

Nicht-Interferenz

Beispiel: Fülle verschlüsselte Nachricht mit Noncen auf:

$$P(x,y) = \nu k. \quad \begin{array}{l} \nu a.\, \mathsf{let}\ m_1 = \mathsf{encrypt}((x,a),k)\ \mathsf{in} \\ \nu b.\, \mathsf{let}\ m_2 = \mathsf{encrypt}((y,b),k)\ \mathsf{in} \\ \overline{c}\langle m_1.m_2 \rangle \end{array}$$

Es gilt $P(M_1, M_2) \approx P(N_1, N_2)$ für alle geschlossenen Terme M_1, M_2, N_1 und N_2.

Äquivalenzbeweise mit Horn-Klauseln

Eine bestimmte Klasse von Prozessäquivalenzen kann man durch Kodierung in Horn-Klauseln und Resolution beweisen.

Idee:

- Prozesse, die sich nur in Geheimnissen unterscheiden, sind syntaktisch sehr ähnlich.
- Zeige, dass man auch unter Reduktion die syntaktische Ähnlichkeit stets erhalten kann.
- Die Äquivalenz der Prozesse folgt dann daraus, dass die ähnlichen Prozesse stets nur auf die gleiche Art reduzieren können.

Biprozesse

Biprozesse sind Prozesse, in denen neue Terme $\mathsf{diff}[M, M']$ und neue Prozesse $\mathsf{diff}[P, P']$ vorkommen können.

$\mathsf{fst}(P)$ ist der Prozess, den man aus P erhält, indem $\mathsf{diff}[M, M']$ durch M ersetzt wird und $\mathsf{diff}[P, P']$ durch P. (analog: $\mathsf{snd}(P)$)

Ziel: Beweise Äquivalenzen der Form $\mathsf{fst}(P) \approx \mathsf{snd}(P)$ beweisen.

Beispiel:

$$
\begin{aligned}
P = \nu k. \quad & \nu a.\, \mathsf{let}\ m_1 = \mathsf{encrypt}((\mathsf{diff}[M, M'], a), k)\ \mathsf{in} \\
& \nu b.\, \mathsf{let}\ m_2 = \mathsf{encrypt}((\mathsf{diff}[N, N'], b), k)\ \mathsf{in} \\
& \overline{c}\langle m_1.m_2 \rangle
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{fst}(P) = \nu k. \quad & \nu a.\, \mathsf{let}\ m_1 = \mathsf{encrypt}((M, a), k)\ \mathsf{in} \\
& \nu b.\, \mathsf{let}\ m_2 = \mathsf{encrypt}((N, b), k)\ \mathsf{in} \\
& \overline{c}\langle m_1.m_2 \rangle
\end{aligned}
$$

Biprozesse

Die Reduktionssemantik eines Biprozesses P ist so definiert, dass $\mathsf{fst}(P)$ und $\mathsf{snd}(P)$ gleichzeitig und *auf die gleiche Art* reduziert werden:

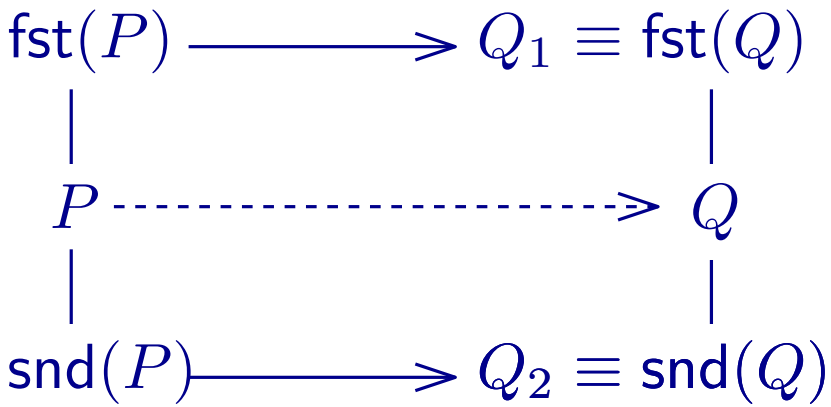
$\overline{N}\langle M\rangle.Q \mid N'(x).P \longrightarrow Q \mid P\{M/x\}$
falls $\mathsf{fst}(N) = \mathsf{fst}(N')$ und $\mathsf{snd}(N) = \mathsf{snd}(N')$

$\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q \longrightarrow P\{\mathsf{diff}[M_1, M_2]/x\}$
falls $\mathsf{fst}(D) \to M_1$ und $\mathsf{snd}(D) \to M_2$

$\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q \longrightarrow Q$
falls es keine M_1 und M_2 gibt, mit $\mathsf{fst}(D) \to M_1$ und $\mathsf{snd}(D) \to M_2$

Biprozesse

Definition: Ein Biprozess ist *uniform* falls gilt:
Aus $\mathsf{fst}(P) \to Q_1$ folgt $P \to Q$ für einen Biprozess Q, für den $\mathsf{fst}(Q) \equiv Q_1$ gilt (und analog mit snd statt fst).



Satz: Sei P ein geschlossener Biprozess. Wenn

$$E[P] \Longrightarrow \equiv Q \quad \Longrightarrow \quad Q \text{ ist uniform}$$

für jeden Auswertungskontext $E[-]$ ohne Vorkommen von diff gilt, dann gilt $\mathsf{fst}(P) \approx \mathsf{snd}(\mathrm{P})$.

Repräsentation durch Klauseln

Die Berechnung eines Biprozesses P wird ähnlich wie normale Prozesse in Horn-Klauseln kodiert.

Die möglichen Aktionen der beiden Komponenten $\mathsf{fst}(P)$ und $\mathsf{snd}(P)$ werden jedoch getrennt verfolgt.

Es werden folgende Prädikate benutzt:

- $\mathsf{att}(p, p')$ — Der Angreifer könnte von $\mathsf{fst}(P)$ Nachricht p bekommen, von $\mathsf{snd}(P)$ die Nachricht p'.
- $\mathsf{mess}(p_1, p_2, p_1', p_2')$ — In $\mathsf{fst}(P)$ könnte die Nachricht p_2 über den Kanal p_1 geschickt werden. (analog für $\mathsf{snd}(P)$)
- $\mathsf{input}(p_1, p_1')$ — In $\mathsf{fst}(P)$ ist eine Eingabe auf Kanal p_1 möglich. (analog für $\mathsf{snd}(P)$)
- $\mathsf{nounif}(p, p')$ — Es ist unmöglich p und p' zu unifizieren.
- bad — Ein nichtuniformer Biprozess ist erreichbar.

Beispiel

$$P(x,y) = \nu k. \quad \begin{array}{l} \text{let } m_1 = \text{encrypt}(x,k) \text{ in} \\ \text{let } m_2 = \text{encrypt}(y,k) \text{ in} \\ \overline{c}\langle m_1.m_2 \rangle \end{array}$$

Klausel für P:

$$\supset \text{msg}(c, (\text{encrypt}(x,k), \text{encrypt}(y,k)), c, (\text{encrypt}(x',k), \text{encrypt}(y',k)))$$

Beispiel für allgemeine Klausel:

$$\text{att}(u,v) \wedge \text{att}(u,v') \wedge \text{nounif}(v,v') \supset \text{bad}$$

Angreifer kann schließen auf:

$\text{att}(\text{encrypt}(x,k), \text{encrypt}(x',k))$ und $\text{att}(\text{encrypt}(y,k), \text{encrypt}(y',k))$.

Mit $u \mapsto \text{encrypt}(x,k), y \mapsto x$ und $\text{nounif}(\text{encrypt}(x',k), \text{encrypt}(y',k))$ folgt bad.