## DAT6 Projekteksamen

# Type Checking Versus Flow Logics

## relations between static analysis methods for cryptographic protocols

Projektgruppe d622a

Lars H. Jensen and Bjørn M. Vester

Institut for Datalogi

Aalborg Universitet

# Introduktion

# Introduktion

- Bevise autenticitets egenskaber.

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:
  - LySA er Turing-komplet.

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:
  - LySA er Turing-komplet.
  - Autenticitet er en ikke-triviel egenskab.

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:
  - LySA er Turing-komplet.
  - Autenticitet er en ikke-triviel egenskab.
  - Ingen algoritmisk metode til at afgøre denne egenskab.

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:
  - LySA er Turing-komplet.
  - Autenticitet er en ikke-triviel egenskab.
  - Ingen algoritmisk metode til at afgøre denne egenskab.
- Problemer på miljøet:

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:
  - LySA er Turing-komplet.
  - Autenticitet er en ikke-triviel egenskab.
  - Ingen algoritmisk metode til at afgøre denne egenskab.
- Problemer på miljøet:
  - Miljøet kan være fjendtligt.

# Introduktion

- Bevise autenticitets egenskaber.
- Betragter egenskaber for processer i LySA.
- Uafgørbarheds problemer:
  - LySA er Turing-komplet.
  - Autenticitet er en ikke-triviel egenskab.
  - Ingen algoritmisk metode til at afgøre denne egenskab.
- Problemer på miljøet:
  - Miljøet kan være fjendtligt.
  - Egenskaberne for en proces skal holde i alle mulige miljøer.

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.
  - Metoder baseret på BAN logikker.

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.
  - Metoder baseret på BAN logikker.
  - Model check.

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.
  - Metoder baseret på BAN logikker.
  - Model check.
- Fokus i vores arbejde:

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.
  - Metoder baseret på BAN logikker.
  - Model check.
- Fokus i vores arbejde:
  - Kontrol flow analyse.

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.
  - Metoder baseret på BAN logikker.
  - Model check.
- Fokus i vores arbejde:
  - Kontrol flow analyse.
  - Type systemer.

# Metoderne til analyse af sikkerheds egenskaber

- Relateret arbejde:
  - Ækvivalens test gennem bisimulering.
  - Metoder baseret på BAN logikker.
  - Model check.
- Fokus i vores arbejde:
  - Kontrol flow analyse.
  - Type systemer.
  - Er der en relation mellem disse metoder?

# Oversigt af præsentationen

# Oversigt af præsentationen

- Kontrol flow analyse

# Oversigt af præsentationen

- Kontrol flow analyse
- Type system

# Oversigt af præsentationen

- Kontrol flow analyse
- Type system
- Oversættelse fra LySA til Typed LySA

# Oversigt af præsentationen

- Kontrol flow analyse
- Type system
- Oversættelse fra LySA til Typed LySA
- Type inferens

# Oversigt af præsentationen

- Kontrol flow analyse
- Type system
- Oversættelse fra LySA til Typed LySA
- Type inferens
- Kontrol flow analyse med korrespondancer

# Oversigt af præsentationen

- Kontrol flow analyse
- Type system
- Oversættelse fra LySA til Typed LySA
- Type inferens
- Kontrol flow analyse med korrespondancer
- Konklusion

# Kontrol flow analyse

- Formaliseret af C. Bodei, M. Buchholtz, P. Degano, F. Nielson og H. Riis Nielson i *Automatic Validation of Protocol Narration*.

# Kontrol flow analyse

- Formaliseret af C. Bodei, M. Buchholtz, P. Degano, F. Nielson og H. Riis Nielson i *Automatic Validation of Protocol Narration*.
- Forudsiger statisk hvilke værdier og opførsel der opstår dynamisk

# Kontrol flow analyse

- Formaliseret af C. Bodei, M. Buchholtz, P. Degano, F. Nielson og H. Riis Nielson i *Automatic Validation of Protocol Narration*.
- Forudsiger statisk hvilke værdier og opførsel der opstår dynamisk
- Giver *sikre approksimative* svar.

# Kontrol flow analyse

- Formaliseret af C. Bodei, M. Buchholtz, P. Degano, F. Nielson og H. Riis Nielson i *Automatic Validation of Protocol Narration*.
- Forudsiger statisk hvilke værdier og opførsel der opstår dynamisk
- Giver *sikre approksimative* svar.
- Enver kryptografisk operation er annoteret med et *kryto-punkt* $\ell$, hvor $\ell \in \mathcal{C}$.

# Kontrol flow analyse

- Formaliseret af C. Bodei, M. Buchholtz, P. Degano, F. Nielson og H. Riis Nielson i *Automatic Validation of Protocol Narration*.
- Forudsiger statisk hvilke værdier og opførsel der opstår dynamisk
- Giver *sikre approksimative* svar.
- Enver kryptografisk operation er annoteret med et *kryto-punkt* $\ell$, hvor $\ell \in \mathcal{C}$.
- Enver kryptering er annoteret med en mængde af destinationer $[\text{dest } \mathcal{L}]$, hvor $\mathcal{L} \subseteq \mathcal{C}$.

# Kontrol flow analyse

- Formaliseret af C. Bodei, M. Buchholtz, P. Degano, F. Nielson og H. Riis Nielson i *Automatic Validation of Protocol Narration*.
- Forudsiger statisk hvilke værdier og opførsel der opstår dynamisk
- Giver *sikre approksimative* svar.
- Enver kryptografisk operation er annoteret med et *kryto-punkt* $\ell$, hvor $\ell \in \mathcal{C}$.
- Enver kryptering er annoteret med en mængde af destinationer $[\text{dest } \mathcal{L}]$, hvor $\mathcal{L} \subseteq \mathcal{C}$.
- Enver dekryptering er annoteret med en mængde af udgangspunkter $[\text{orig } \mathcal{L}]$.

# Resultat af en kontrol flow analyse

- En *løsning* består af 3 mængder:

# Resultat af en kontrol flow analyse

- En *løsning* består af 3 mængder:
  1. En mængde af værdier en variable kan være bundet til.

$$\rho : \lfloor \mathcal{X} \rfloor \rightarrow \wp(\mathcal{V})$$

# Resultat af en kontrol flow analyse

- En *løsning* består af 3 mængder:
  1. En mængde af værdier en variable kan være bundet til.

$$\rho : \lfloor \mathcal{X} \rfloor \to \wp(\mathcal{V})$$

  2. Mængden af beskeder som kan blive sendt på æteren.

$$\kappa \subseteq \wp(\mathcal{V}^*)$$

# Resultat af en kontrol flow analyse

- En *løsning* består af 3 mængder:
  1. En mængde af værdier en variable kan være bundet til.

$$\rho : \lfloor \mathcal{X} \rfloor \to \wp(\mathcal{V})$$

  2. Mængden af beskeder som kan blive sendt på æteren.

$$\kappa \subseteq \wp(\mathcal{V}^*)$$

  3. Mængden af *krypto-punkts overtrædelser*:

$$\psi \subseteq \mathcal{C} \times \mathcal{C}$$

# Konstruktion af en angriber

- En *angriber* kan kun bruge de følgende annoteringer.

# Konstruktion af en angriber

- En *angriber* kan kun bruge de følgende annoteringer.
  - Alle kryptografiske operationer har krypto-punkt $\ell_\bullet$.

# Konstruktion af en angriber

- En *angriber* kan kun bruge de følgende annoteringer.
  - Alle kryptografiske operationer har krypto-punkt $\ell_\bullet$.
  - Alle destination og udgangspunkt mængder er $\mathcal{C}$.

# Konstruktion af en angriber

- En *angriber* kan kun bruge de følgende annoteringer.
  - Alle kryptografiske operationer har krypto-punkt $\ell_\bullet$.
  - Alle destination og udgangspunkt mængder er $\mathcal{C}$.
- Protokoller kan blive eksekveret parallelt med en tilfældig angriber $P_\bullet$:

$$P \mid P_\bullet$$

# Konstruktion af en angriber

- En *angriber* kan kun bruge de følgende annoteringer.
  - Alle kryptografiske operationer har krypto-punkt $\ell_\bullet$.
  - Alle destination og udgangspunkt mængder er $\mathcal{C}$.
- Protokoller kan blive eksekveret parallelt med en tilfældig angriber $P_\bullet$:

$$P \mid P_\bullet$$

- Angriberen $P_\bullet$ er modelleret af formlen $\mathcal{F}^{\mathsf{DY}}$ som fanger alle Dolev-Yao egenskaber.

# Konstruktion af en angriber

- En *angriber* kan kun bruge de følgende annoteringer.
  - Alle kryptografiske operationer har krypto-punkt $\ell_\bullet$.
  - Alle destination og udgangspunkt mængder er $\mathcal{C}$.
- Protokoller kan blive eksekveret parallelt med en tilfældig angriber $P_\bullet$:

$$P \mid P_\bullet$$

- Angriberen $P_\bullet$ er modelleret af formlen $\mathcal{F}^{\mathsf{DY}}$ som fanger alle Dolev-Yao egenskaber.
- Hvis $(\rho, \kappa, \psi)$ tilfredsstiller $\mathcal{F}^{\mathsf{DY}}$, så er $(\rho, \kappa) \models P_\bullet : \psi$ for alle angribere $P_\bullet$.

# Egenskaber for kontrol flow analyse

- Definition: En proces $P$ garanterer *statisk autenticitet* hvis der eksisterer $(\rho, \kappa)$ således at $(\rho, \kappa) \models P : \emptyset$ og $(\rho, \kappa, \emptyset)$ tilfredsstiller $\mathcal{F}^{\mathsf{DY}}$.

# Egenskaber for kontrol flow analyse

- Definition: En proces $P$ garanterer *statisk autenticitet* hvis der eksisterer $(\rho, \kappa)$ således at $(\rho, \kappa) \models P : \emptyset$ og $(\rho, \kappa, \emptyset)$ tilfredsstiller $\mathcal{F}^{\mathsf{DY}}$.
- Sætning: En proces $P$ garanterer *dynamisk autenticitet* med hensyn til annoteringerne i $P$ hvis og kun hvis reference overvågningsenheden ikke kan aborte $P \mid P_\bullet$ uanset valg af $P_\bullet$.

# Egenskaber for kontrol flow analyse

- Definition: En proces $P$ garanterer *statisk autenticitet* hvis der eksisterer $(\rho, \kappa)$ således at $(\rho, \kappa) \models P : \emptyset$ og $(\rho, \kappa, \emptyset)$ tilfredsstiller $\mathcal{F}^{\mathsf{DY}}$.
- Sætning: En proces $P$ garanterer *dynamisk autenticitet* med hensyn til annoteringerne i $P$ hvis og kun hvis reference overvågningsenheden ikke kan aborte $P \mid P_\bullet$ uanset valg af $P_\bullet$.
- Sætning: Hvis $P$ garanterer statisk autenticitet så garanterer $P$ dynamisk autenticitet.

# Type system

# Introduktion til typesystemer

- Type og effekt systemer til verificering af autenticitets egenskaber som foreslået af Gordon og Jeffrey i *Authenticity by Typing* (2001).

# Introduktion til typesystemer

- Type og effekt systemer til verificering af autenticitets egenskaber som foreslået af Gordon og Jeffrey i *Authenticity by Typing* (2001).
- Syntaks af LySA er udvidet til at inkludere hændelser på formen begin!($\vec{n}$) og end($\vec{n}$).

# Introduktion til typesystemer

- Type og effekt systemer til verificering af autenticitets egenskaber som foreslået af Gordon og Jeffrey i *Authenticity by Typing* (2001).
- Syntaks af LySA er udvidet til at inkludere hændelser på formen begin!($\vec{n}$) og end($\vec{n}$).
- En proces er *sikker* hvis og kun hvis for enhver kørsel da er enhver end($\vec{n}$) hændelse forudgået af en begin!($\vec{n}$) hændelse.

# Introduktion til typesystemer

- Type og effekt systemer til verificering af autenticitets egenskaber som foreslået af Gordon og Jeffrey i *Authenticity by Typing* (2001).
- Syntaks af LySA er udvidet til at inkludere hændelser på formen begin!($\vec{n}$) og end($\vec{n}$).
- En proces er *sikker* hvis og kun hvis for enhver kørsel da er enhver end($\vec{n}$) hændelse forudgået af en begin!($\vec{n}$) hændelse.
- Typer for nøgler har en mængde *skjulte effekter* på formen !begun($\vec{n}$).

# Introduktion til typesystemer

- Type og effekt systemer til verificering af autenticitets egenskaber som foreslået af Gordon og Jeffrey i *Authenticity by Typing* (2001).
- Syntaks af LySA er udvidet til at inkludere hændelser på formen begin!($\vec{n}$) og end($\vec{n}$).
- En proces er *sikker* hvis og kun hvis for enhver kørsel da er enhver end($\vec{n}$) hændelse forudgået af en begin!($\vec{n}$) hændelse.
- Typer for nøgler har en mængde *skjulte effekter* på formen !begun($\vec{n}$).
- Ved krypteringer skal skjulte effekter være gældende for at de er lovlige.

# Introduktion til typesystemer

- Type og effekt systemer til verificering af autenticitets egenskaber som foreslået af Gordon og Jeffrey i *Authenticity by Typing* (2001).
- Syntaks af LySA er udvidet til at inkludere hændelser på formen begin!($\vec{n}$) og end($\vec{n}$).
- En proces er *sikker* hvis og kun hvis for enhver kørsel da er enhver end($\vec{n}$) hændelse forudgået af en begin!($\vec{n}$) hændelse.
- Typer for nøgler har en mængde *skjulte effekter* på formen !begun($\vec{n}$).
- Ved krypteringer skal skjulte effekter være gældende for at de er lovlige.
- Ved dekrypteringer giver skjulte effekter lov til senere hændelser.

# Egenskaber for type systemet

- Definition af robust sikkerhed: En proces $P$ er *robust sikker* hvis og kun hvis $(P \mid O)$ er sikker for enhver angriber proces $O$.

# Egenskaber for type systemet

- Definition af robust sikkerhed: En proces $P$ er *robust sikker* hvis og kun hvis $(P \mid O)$ er sikker for enhver angriber proces $O$.
- Sætning (Safety): Hvis $(E \vdash P)$ da er $P$ sikker.

# Egenskaber for type systemet

- Definition af robust sikkerhed: En proces $P$ er *robust sikker* hvis og kun hvis $(P \mid O)$ er sikker for enhver angriber proces $O$.
- Sætning (Safety): Hvis $(E \vdash P)$ da er $P$ sikker.
- Sætning (Robust Sikkerhed): Hvis $n_1 : \mathit{Un}\ , \cdots , n_k : \mathit{Un} \vdash P)$, da er $P$ robust sikker.

# Oversættelse fra LySA til Typed LySA

# Oversættelse

- Vi ønsker at oversætte fra LySA til Typed LySA og bevare sikkerheds egenskaberne.

# Oversættelse

- Vi ønsker at oversætte fra LySA til Typed LySA og bevare sikkerheds egenskaberne.

**Indkodnings funktion $[\![\cdot]\!]$:**

$$[\![P]\!] \triangleq \text{begin!}(o_1, d_1).\cdots.\text{begin!}(o_n, d_n).(\!|P|\!)$$

**where** $\text{AC}(P) = \{(o_1, d_1), \ldots, (o_n, d_n)\}$

# Oversættelse (2)

**Hjælpe relation** $\mathsf{AC}(P)$**:**

$$
\begin{aligned}
\mathsf{AC}(n) &\triangleq \emptyset \\
\mathsf{AC}(x) &\triangleq \emptyset \\
\mathsf{AC}(\{M_1, \ldots, M_k\}^o_{M_0}[\mathsf{dest}\ D]) &\triangleq (o \times D) \cup \mathsf{AC}(M_1) \cup \cdots \cup \mathsf{AC}(M_k) \\
\mathsf{AC}(0) &\triangleq \emptyset \\
\mathsf{AC}(\langle M_1, \ldots, M_k \rangle.P) &\triangleq \mathsf{AC}(M_1) \cup \cdots \cup \mathsf{AC}(M_k) \cup \mathsf{AC}(P) \\
\mathsf{AC}((M_1, \ldots, M_j; x_{j+1}, \ldots, x_k).P) &\triangleq \mathsf{AC}(P) \\
\vdots &\triangleq \vdots
\end{aligned}
$$

# Oversættelse (3)

**Indkodnings algoritme $(\!|\cdot|\!)$:**

$$\vdots \triangleq \vdots$$

$$\left.\begin{array}{r}(\!|\{M_1,\ldots,M_k\}^{o}_{M_0}|\!) \\ [\text{dest } \{d_0,\ldots,d_n\}]\end{array}\right\} \triangleq \{(\!|M_1|\!),\ldots,(\!|M_k|\!),o\}_{(\!|M_0|\!)}$$

$$\vdots \triangleq \vdots$$

$$\left.\begin{array}{r}(\!|\text{decrypt } M \text{ as } \{M_1,\ldots,M_j;\\ x_{j+1},\ldots,x_k\}^{d}_{M_0}|\!) \\ [\text{orig } \{o_1,\ldots,o_n\}] \text{ in } P|\!)\end{array}\right\} \triangleq \left\{\begin{array}{l}\text{decrypt } (\!|M|\!) \text{ as } \{(\!|M_1|\!),\ldots,(\!|M_j|\!);\\ \{x_{j+1},\ldots,x_k,y\}_{(\!|M_0|\!)} \text{ in end}(y,d).(\!|P|\!)\end{array}\right.$$

# Egenskaber ved oversættelsen

- Antagelse: kun navne er brugt i pattern matches.

# Egenskaber ved oversættelsen

- Antagelse: kun navne er brugt i pattern matches.
- Sætning: En proces $Q$ med velformede krypto-punkter der overholder ovenstående antagelse garanterer dynamisk autenticitet hvis og kun hvis $[\![Q]\!]$ er sikker.

# Type inferens på oversættelsen

# Type inferens

- Hvis en kontrol flow analyse siger en LySA proces er sikker kan vi så få type systemet til at sige det samme til en oversat proces i Typed LySA?

# Type inferens

- Hvis en kontrol flow analyse siger en LySA proces er sikker kan vi så få type systemet til at sige det samme til en oversat proces i Typed LySA?
- Oversættelse satte typen på alle navne til Un.

# Type inferens

- Hvis en kontrol flow analyse siger en LySA proces er sikker kan vi så få type systemet til at sige det samme til en oversat proces i Typed LySA?
- Oversættelse satte typen på alle navne til Un.
- Type inferens forsøger at erstatte Un med en type så vi kan replikere sikkerheds undersøgelsen.

# Begrænsninger

- Beskeder skal have samme antal elementer hvis de krypteres med den samme nøgle.

# Begrænsninger

- Beskeder skal have samme antal elementer hvis de krypteres med den samme nøgle.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ på plads $i$ i en besked da må $K_1$ aldrig bruges til at kryptere andet end $K_2$ på plads $i$.

# Begrænsninger

- Beskeder skal have samme antal elementer hvis de krypteres med den samme nøgle.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ på plads $i$ i en besked da må $K_1$ aldrig bruges til at kryptere andet end $K_2$ på plads $i$.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ enten direkte eller via lagdelt kryptering, da må $K_2$ aldrig bruges til at kryptere $K_1$ enten direkte eller gennem lagdelt kryptering.

# Begrænsninger

- Beskeder skal have samme antal elementer hvis de krypteres med den samme nøgle.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ på plads $i$ i en besked da må $K_1$ aldrig bruges til at kryptere andet end $K_2$ på plads $i$.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ enten direkte eller via lagdelt kryptering, da må $K_2$ aldrig bruges til at kryptere $K_1$ enten direkte eller gennem lagdelt kryptering.
- Kun navne må bruges som nøgler.

# Begrænsninger

- Beskeder skal have samme antal elementer hvis de krypteres med den samme nøgle.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ på plads $i$ i en besked da må $K_1$ aldrig bruges til at kryptere andet end $K_2$ på plads $i$.
- Hvis en nøgle $K_1$ bruges til at kryptere $K_2$ enten direkte eller via lagdelt kryptering, da må $K_2$ aldrig bruges til at kryptere $K_1$ enten direkte eller gennem lagdelt kryptering.
- Kun navne må bruges som nøgler.
- Alle navne kan flyde til alle variabler.

# Hjælpe relationer (1)

**Konstruktion af hjælpe funktioner $\mathcal{K}_{enc}, \mathcal{K}_{dest}$ (Given $\rho$):**

(Rel $\mathcal{K}_{enc}, \mathcal{K}_{dest}$ Enc)

$$\frac{\mathcal{K}_{enc}, \mathcal{K}_{dest}, M_1 \rightarrow \mathcal{K}^1_{enc}, \mathcal{K}^1_{dest} \quad \cdots \quad \mathcal{K}^{k-1}_{enc}, \mathcal{K}^{k-1}_{dest}, M_k \rightarrow \mathcal{K}^k_{enc}, \mathcal{K}^k_{dest}}{\mathcal{K}_{enc}, \mathcal{K}_{dest}, \{M_1, \ldots, M_k\}_{M_0} \rightarrow \mathcal{K}^k_{enc}[v_1 \overset{+}{\longmapsto} (\overline{v_1} \times \cdots \times \overline{v_k}), \ldots, v_i \overset{+}{\longmapsto} (\overline{v_1} \times \cdots \times \overline{v_k})], \mathcal{K}^k_{dest}}$$

where $\rho(\lfloor M_0 \rfloor) = \{v_1, \ldots, v_i\} \wedge \rho(\lfloor M_1 \rfloor) = \overline{v_1} \quad \cdots \quad \rho(\lfloor M_k \rfloor) = \overline{v_k}$

# Hjælpe relationer (2)

**Konstruktion af hjælpe funktioner $\mathcal{K}_{enc}, \mathcal{K}_{dest}$ (Given $\rho$):**

(Rel $\mathcal{K}_{enc}, \mathcal{K}_{dest}$ Decr)

$$\frac{\mathcal{K}_{enc}, \mathcal{K}_{dest}, M \to \mathcal{K}''_{enc}, \mathcal{K}''_{dest} \quad \mathcal{K}''_{enc}, \mathcal{K}''_{dest}, P \to \mathcal{K}'_{enc}, \mathcal{K}'_{dest}}{\begin{array}{c}\mathcal{K}_{enc}, \mathcal{K}_{dest}, \text{ decrypt } M \text{ as } \{M_1, \ldots, M_j; x_{j+1}, \ldots, x_k\}_{M_0} \text{ in} \\ \text{end}(x_k, d).P \to \mathcal{K}'_{enc}, \mathcal{K}'_{dest}[v_1 \stackrel{+}{\longmapsto} d, \ldots, v_i \stackrel{+}{\longmapsto} d] \\ \text{where } \rho(\lfloor M_0 \rfloor) = \{v_1, \ldots, v_i\}\end{array}}$$

# Type inferens (1)

**Konstruktion af $\Gamma$ (given $\mathcal{K}_{enc}, \mathcal{K}_{dest}$):**

(Sol Res Non-key)

$$\frac{\lfloor n \rfloor \notin \mathsf{dom}(\mathcal{K}_{enc}) \quad \Gamma[\lfloor n \rfloor \overset{*}{\mapsto} \mathsf{Un}], P \rightarrow \Gamma'}{\Gamma, (\nu\, n : \mathsf{Un}) P \rightarrow \Gamma'}$$

# Type inferens (2)

**Konstruktion af $\Gamma$ (given $\mathcal{K}_{enc}, \mathcal{K}_{dest}$):**

(Sol Res Key)

$$\frac{\lfloor n \rfloor \in \mathsf{dom}(\mathcal{K}_{enc}) \qquad \Gamma[\lfloor n \rfloor \overset{*}{\mapsto} \mathsf{Key}(x_1 : T'_1, \ldots, x_k : T'_k, x_{k+1} : \mathsf{Un})[\vec{B}]], P \rightarrow \Gamma'}{\Gamma, (\nu\, n : \mathsf{Un})P \rightarrow \Gamma'}$$

where $\mathcal{K}_{dest}(\lfloor n \rfloor) = \{d_1, \ldots, d_n\}$

and $\vec{B} =$ !begun$(x_{k+1}, d_1), \ldots,$!begun$(x_{k+1}, d_n)$

and $\mathcal{K}_{enc}(\lfloor n \rfloor) = \{(v_{1,1}, \ldots, v_{1,k}), \ldots, (v_{j,1}, \ldots, v_{j,k})\}$

and $\forall i \in [1..k] : (v_{i,1} \in \mathsf{dom}(\Gamma) \Rightarrow T'_i = \Gamma(v_{i,1})) \wedge$

$((v_{i,1} \notin \mathsf{dom}(\Gamma) \ \wedge \ v_{i,1} \in \mathcal{N}) \Rightarrow T'_i = H_{v_{i,1}}) \wedge$

$((v_{i,1} \notin \mathsf{dom}(\Gamma) \ \wedge \ v_{i,1} \notin \mathcal{N}) \Rightarrow T'_i = \mathsf{Un})$

# Type inferens

- Definition: Lad $Q$ være en TYPED LYSA proces, da er $Q_\Gamma$ den samme proces, med den undtagelse at alle restriktionerne $(\nu\, n : T)$ er ændret således at $T = \Gamma(\lfloor n \rfloor)$.

# Type inferens

- Definition: Lad $Q$ være en TYPED LYSA proces, da er $Q_\Gamma$ den samme proces, med den undtagelse at alle restriktionerne $(\nu\, n : T)$ er ændret således at $T = \Gamma(\lfloor n \rfloor)$.
- Formodning: Given en LYSA proces $P$ der overholder listede antagelser og har en analyse $(\rho, \kappa, \emptyset)$ således at $(\rho, \kappa) \models P : \emptyset$ og $(\rho, \kappa, \emptyset)$ overholder $\mathcal{F}^{\mathsf{DY}}$ da er $\vec{n} : \mathit{Un} \vdash [\![P]\!]_\Gamma$ hvor $\vec{n} = \mathit{fn}([\![P]\!]_\Gamma)$.

# Kontrol-flow-analyse med Korrespondancer

# Kontrol-flow-analyse med Korrespondancer

- Typecheck har begrænsninger:

# Kontrol-flow-analyse med Korrespondancer

- Typecheck har begrænsninger:
  - Krypteringer af samme nøgle skal have samme længde.

# Kontrol-flow-analyse med Korrespondancer

- Typecheck har begrænsninger:
  - Krypteringer af samme nøgle skal have samme længde.
  - Kan ikke analysere pattern-matches.

# Kontrol-flow-analyse med Korrespondancer

- Typecheck har begrænsninger:
  - Krypteringer af samme nøgle skal have samme længde.
  - Kan ikke analysere pattern-matches.
- Hvordan kan vi omgå disse begrænsninger?

# Kontrol-flow-analyse med Korrespondancer

- Typecheck har begrænsninger:
  - Krypteringer af samme nøgle skal have samme længde.
  - Kan ikke analysere pattern-matches.
- Hvordan kan vi omgå disse begrænsninger?
  - Lav indkodning fra korrespondancer til krypto-punkter.

# Kontrol-flow-analyse med Korrespondancer

- Typecheck har begrænsninger:
  - Krypteringer af samme nøgle skal have samme længde.
  - Kan ikke analysere pattern-matches.
- Hvordan kan vi omgå disse begrænsninger?
  - Lav indkodning fra korrespondancer til krypto-punkter.
  - Lav kontrol-flow-analyse direkte på processen med korrespondancer.

# Opsamling af Effekter

- Begreb om linearitet.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.
- end($n$) checker at !begun($\lfloor n \rfloor$) er i $\psi$.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.
- end($n$) checker at !begun($\lfloor n \rfloor$) er i $\psi$.
- Variabler skal håndteres anderledes.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.
- end($n$) checker at !begun($\lfloor n \rfloor$) er i $\psi$.
- Variabler skal håndteres anderledes.
- Første tilgangsvinkel: end($x$) checker at $\forall V \in \rho(\lfloor x \rfloor)$ : !begun($V$) er i $\psi$.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.
- end($n$) checker at !begun($\lfloor n \rfloor$) er i $\psi$.
- Variabler skal håndteres anderledes.
- Første tilgangsvinkel: end($x$) checker at $\forall V \in \rho(\lfloor x \rfloor)$ : !begun($V$) er i $\psi$.
- Hvordan håndteres variabler i begynd-markeringer? Eksempelvis begin!($x$).

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.
- end($n$) checker at !begun($\lfloor n \rfloor$) er i $\psi$.
- Variabler skal håndteres anderledes.
- Første tilgangsvinkel: end($x$) checker at $\forall V \in \rho(\lfloor x \rfloor)$ : !begun($V$) er i $\psi$.
- Hvordan håndteres variabler i begynd-markeringer? Eksempelvis begin!($x$).
- Løsning: placer !begun($\lfloor x \rfloor$) i $\psi$.

# Opsamling af Effekter

- Begreb om linearitet.
- Fejl-komponenten $\psi$ bliver til beholder for effekter.
- begin!($n$) indsætter !begun($\lfloor n \rfloor$) i $\psi$.
- end($n$) checker at !begun($\lfloor n \rfloor$) er i $\psi$.
- Variabler skal håndteres anderledes.
- Første tilgangsvinkel: end($x$) checker at $\forall V \in \rho(\lfloor x \rfloor)$ : !begun($V$) er i $\psi$.
- Hvordan håndteres variabler i begynd-markeringer? Eksempelvis begin!($x$).
- Løsning: placer !begun($\lfloor x \rfloor$) i $\psi$.
- Anden tilgangsvinkel: end($x$) checker at !begun($\lfloor x \rfloor$) er i $\psi$.

# Overførsel af Effekter: Afsender

- Første tilgangsvinkel: annoter nøgle på samme måde som typesystemet.

# Overførsel af Effekter: Afsender

- Første tilgangsvinkel: annoter nøgle på samme måde som typesystemet.
- Problem ved brug af flere korrespondancer på samme nøgle:

$A$ : begin$(A, B)$

$A \rightarrow B$ : $\{M_1\}_K$

$B$ : end$(A, B)$

$A$ : begin$(A, C)$

$A \rightarrow C$ : $\{M_2\}_K$

$C$ : end$(A, C)$

# Overførsel af Effekter: Afsender

- Første tilgangsvinkel: annoter nøgle på samme måde som typesystemet.
- Problem ved brug af flere korrespondancer på samme nøgle:

$$
\begin{array}{ll}
A: & \mathsf{begin}(A, B) \\
A \rightarrow B: & \{M_1\}_K \\
B: & \mathsf{end}(A, B) \\
A: & \mathsf{begin}(A, C) \\
A \rightarrow C: & \{M_2\}_K \\
C: & \mathsf{end}(A, C)
\end{array}
$$

- Problem: vi kan ikke finde en annotering til $K$.

# Overførsel af Effekter: Afsender

- Første tilgangsvinkel: annoter nøgle på samme måde som typesystemet.
- Problem ved brug af flere korrespondancer på samme nøgle:

$$
\begin{array}{ll}
A: & \text{begin}(A, B) \\
A \rightarrow B: & \{M_1\}_K \\
B: & \text{end}(A, B) \\
A: & \text{begin}(A, C) \\
A \rightarrow C: & \{M_2\}_K \\
C: & \text{end}(A, C)
\end{array}
$$

- Problem: vi kan ikke finde en annotering til $K$.
- Løsning: Effekter overføres gennem annoteringer på krypteringer.

# Overførsel af Effekter: Afsender

- Første tilgangsvinkel: annoter nøgle på samme måde som typesystemet.
- Problem ved brug af flere korrespondancer på samme nøgle:

$A$ : begin$(A, B)$

$A \rightarrow B$ : $\{M_1\}_K$[!begun$(A, B)$]

$B$ : end$(A, B)$

$A$ : begin$(A, C)$

$A \rightarrow C$ : $\{M_2\}_K$[!begun$(A, B)$,!begun$(A, C)$]

$C$ : end$(A, C)$

- Problem: vi kan ikke finde en annotering til $K$.
- Løsning: Effekter overføres gennem annoteringer på krypteringer.

# Overførsel af Effekter: Modtager

- Antag vi kan modtage følgende krypteringer:
  $\{M_1\}_K$[!begun$(A)$] og
  $\{M_2\}_K$[!begun$(A)$, !begun$(B)$]

# Overførsel af Effekter: Modtager

- Antag vi kan modtage følgende krypteringer:
  $\{M_1\}_K$[!begun$(A)$] og
  $\{M_2\}_K$[!begun$(A)$, !begun$(B)$]
- Vi overfører kun fællesmængden af effekter, dvs. !begun$(A)$.

# Overførsel af Effekter: Modtager

- Antag vi kan modtage følgende krypteringer:
  $\{M_1\}_K$[!begun$(A)$] og
  $\{M_2\}_K$[!begun$(A)$, !begun$(B)$]
- Vi overfører kun fællesmængden af effekter, dvs. !begun$(A)$.
- Antag at $K$ er kendt af en angriber.

# Overførsel af Effekter: Modtager

- Antag vi kan modtage følgende krypteringer:
  $\{M_1\}_K$[!begun$(A)$] og
  $\{M_2\}_K$[!begun$(A)$, !begun$(B)$]
- Vi overfører kun fællesmængden af effekter, dvs. !begun$(A)$.
- Antag at $K$ er kendt af en angriber.
- Så vil $\{M_\bullet\}_K[]$ også kunne modtages.

# Overførsel af Effekter: Modtager

- Antag vi kan modtage følgende krypteringer:
  $\{M_1\}_K$[!begun$(A)$] og
  $\{M_2\}_K$[!begun$(A)$, !begun$(B)$]
- Vi overfører kun fællesmængden af effekter, dvs. !begun$(A)$.
- Antag at $K$ er kendt af en angriber.
- Så vil $\{M_\bullet\}_K[]$ også kunne modtages.
- Fællesmængden er da tom.

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

$$P_A \triangleq \text{begin!}(x).\langle \{x\}_K \rangle$$

$$P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$$

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

  $P_A \triangleq \text{begin!}(x).\langle\{x\}_K\rangle$

  $P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$

- Problem: det nytter ikke at overføre !begun$(\lfloor x \rfloor)$ da $x$ bliver bundet som $z$.

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

  $P_A \triangleq \text{begin!}(x).\langle\{x\}_K\rangle$

  $P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$

- Problem: det nytter ikke at overføre !begun$(\lfloor x \rfloor)$ da $x$ bliver bundet som $z$.
- Løsning i typesystemet: $T_K \triangleq \text{Key}(x_1 : \text{Un})[\text{!begun}(x_1)]$.

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

  $P_A \triangleq \text{begin!}(x).\langle\{x\}_K\rangle$

  $P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$

- Problem: det nytter ikke at overføre !begun$(\lfloor x \rfloor)$ da $x$ bliver bundet som $z$.
- Løsning i typesystemet: $T_K \triangleq \text{Key}(x_1 : \text{Un})[\text{!begun}(x_1)]$.
- Løsning i analysen: annoter med $[\text{!begun}(\$1)]$.

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

  $P_A \triangleq \text{begin!}(x).\langle\{x\}_K\rangle$

  $P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$
- Problem: det nytter ikke at overføre !begun$(\lfloor x \rfloor)$ da $x$ bliver bundet som $z$.
- Løsning i typesystemet: $T_K \triangleq \text{Key}(x_1 : \text{Un})[\text{!begun}(x_1)]$.
- Løsning i analysen: annoter med $[\text{!begun}(\$1)]$.
- Eksempel for $\rho(\lfloor y \rfloor)$:
  $\{M_1\}_K[\text{!begun}(\$1)]$ og $\{M_2\}_K[\text{!begun}(\$1)]$.

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

  $P_A \triangleq \text{begin!}(x).\langle\{x\}_K\rangle$

  $P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$
- Problem: det nytter ikke at overføre !begun$(\lfloor x \rfloor)$ da $x$ bliver bundet som $z$.
- Løsning i typesystemet: $T_K \triangleq \text{Key}(x_1 : \text{Un})[\text{!begun}(x_1)]$.
- Løsning i analysen: annoter med $[\text{!begun}(\$1)]$.
- Eksempel for $\rho(\lfloor y \rfloor)$:
  $\{M_1\}_K[\text{!begun}(\$1)]$ og $\{M_2\}_K[\text{!begun}(\$1)]$.
- Fællesmængden af annoteringer er da: !begun$(\lfloor \$1 \rfloor)$.

# Overførsel af Effekter med Variabler

- Hvordan overføres effekter med variabler?
- Betragt følende processer:

  $P_A \triangleq \text{begin!}(x).\langle\{x\}_K\rangle$

  $P_B \triangleq (;y).\text{decrypt } y \text{ as } \{;z\} \text{ in end}(z)$
- Problem: det nytter ikke at overføre !begun$(\lfloor x \rfloor)$ da $x$ bliver bundet som $z$.
- Løsning i typesystemet: $T_K \triangleq \text{Key}(x_1 : \text{Un})[\text{!begun}(x_1)]$.
- Løsning i analysen: annoter med $[\text{!begun}(\$1)]$.
- Eksempel for $\rho(\lfloor y \rfloor)$:
  $\{M_1\}_K[\text{!begun}(\$1)]$ og $\{M_2\}_K[\text{!begun}(\$1)]$.
- Fællesmængden af annoteringer er da: !begun$(\lfloor \$1 \rfloor)$.
- Vi laver en substituering og overfører !begun$(\lfloor z \rfloor)$.

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

  (CFAC Encryption)

$$\frac{\begin{array}{c}\wedge_{i=0}^{k}\,(\rho,\psi) \models M_i : v_i \wedge \\ \forall V_0, V_1, \ldots, V_k : \wedge_{i=0}^{k} V_i \in v_i \Rightarrow \{V_1, \ldots, V_k\}_{V_0}[\psi'] \in v \wedge \\ \mathsf{fv}(\psi') = \emptyset \wedge \psi' \subseteq \psi[\lfloor M_0 \rfloor \mapsto [\![V_0]\!], \ldots, \lfloor M_k \rfloor \mapsto [\![V_k]\!]]\end{array}}{(\rho,\psi) \models \{M_1, \ldots, M_k\}_{M_0} : v}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

  (CFAC Encryption)

$$\frac{\wedge_{i=0}^{k}\,(\rho,\psi) \models M_i : v_i \wedge \forall V_0, V_1, \ldots, V_k : \wedge_{i=0}^{k} V_i \in v_i \Rightarrow \{V_1, \ldots, V_k\}_{V_0}[\psi'] \in v \wedge \mathsf{fv}(\psi') = \emptyset \wedge \psi' \subseteq \psi[\lfloor M_0 \rfloor \mapsto [\![V_0]\!], \ldots, \lfloor M_k \rfloor \mapsto [\![V_k]\!]]}{(\rho,\psi) \models \{M_1, \ldots, M_k\}_{M_0} : v}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

  (CFAC Encryption)

$$\frac{\begin{array}{c}\wedge_{i=0}^{k}\,(\rho,\psi) \models M_i : v_i \wedge \\ \forall V_0, V_1, \ldots, V_k : \wedge_{i=0}^{k} V_i \in v_i \Rightarrow \{V_1, \ldots, V_k\}_{V_0}[\psi'] \in v \wedge \\ \mathsf{fv}(\psi') = \emptyset \wedge \psi' \subseteq \psi[\lfloor M_0 \rfloor \mapsto [\![V_0]\!], \ldots, \lfloor M_k \rfloor \mapsto [\![V_k]\!]]\end{array}}{(\rho,\psi) \models \{M_1, \ldots, M_k\}_{M_0} : v}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

  (CFAC Encryption)

$$\frac{\wedge_{i=0}^{k}\,(\rho,\psi) \models M_i : v_i \wedge \forall V_0, V_1, \ldots, V_k : \wedge_{i=0}^{k} V_i \in v_i \Rightarrow \{V_1, \ldots, V_k\}_{V_0}[\psi'] \in v \wedge \mathsf{fv}(\psi') = \emptyset \wedge \psi' \subseteq \psi[\lfloor M_0 \rfloor \mapsto [\![V_0]\!], \ldots, \lfloor M_k \rfloor \mapsto [\![V_k]\!]]}{(\rho,\psi) \models \{M_1, \ldots, M_k\}_{M_0} : v}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

  (CFAC Encryption)

$$\frac{\begin{array}{c}\wedge_{i=0}^{k}\,(\rho,\psi) \models M_i : v_i \wedge \\ \forall V_0, V_1, \ldots, V_k : \wedge_{i=0}^{k} V_i \in v_i \Rightarrow \{V_1, \ldots, V_k\}_{V_0}[\psi'] \in v \wedge \\ \mathrm{fv}(\psi') = \emptyset \wedge \psi' \subseteq \psi[\lfloor M_0 \rfloor \mapsto [\![V_0]\!], \ldots, \lfloor M_k \rfloor \mapsto [\![V_k]\!]]\end{array}}{(\rho,\psi) \models \{M_1, \ldots, M_k\}_{M_0} : v}$$

- F.eks. bliver $\{A\}_K[!\mathrm{begun}(\$1)]$ til $\{A\}_K[!\mathrm{begun}(A)]$.

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

(CFAC Decrypt)

$$\frac{\begin{array}{c}(\rho,\psi) \models M : v \ \wedge_{i=0}^{j} \ (\rho,\psi) \models M_i : v_i \ \wedge \ (\rho,\kappa) \models P : \psi \cup \psi' \ \wedge \\ \forall \{V_1,\ldots,V_k\}_{V_0}[\psi''] \in v : \ \wedge_{i=0}^{j} V_i \ \mathsf{E} \ v_i \Rightarrow \\ (\wedge_{i=j+1}^{k} \ V_i \in \rho(\lfloor x_i \rfloor) \ \wedge \\ \psi' \subseteq \psi'' \ \cup \ \psi''[[\![V_{j+1}]\!] \mapsto \lfloor x_{j+1} \rfloor, \ldots, [\![V_k]\!] \mapsto \lfloor x_k \rfloor])\end{array}}{(\rho,\kappa) \models \text{ decrypt } M \text{ as } \{M_1,\ldots,M_j; x_{j+1},\ldots,x_k\}_{M_0} \text{ in } P : \psi}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

(CFAC Decrypt)

$$\frac{\begin{array}{c}(\rho,\psi) \models M : v \ \wedge_{i=0}^{j} \ (\rho,\psi) \models M_i : v_i \ \wedge \ (\rho,\kappa) \models P : \psi \cup \psi' \ \wedge \\ \forall \{V_1,\ldots,V_k\}_{V_0}[\psi''] \in v : \ \wedge_{i=0}^{j} V_i \ \mathsf{E} \ v_i \Rightarrow \\ (\wedge_{i=j+1}^{k} \ V_i \in \rho(\lfloor x_i \rfloor) \ \wedge \\ \psi' \subseteq \psi'' \ \cup \ \psi''[[\![V_{j+1}]\!] \mapsto \lfloor x_{j+1} \rfloor, \ldots, [\![V_k]\!] \mapsto \lfloor x_k \rfloor])\end{array}}{(\rho,\kappa) \models \text{ decrypt } M \text{ as } \{M_1,\ldots,M_j; x_{j+1},\ldots,x_k\}_{M_0} \text{ in } P : \psi}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

(CFAC Decrypt)

$$\frac{\begin{array}{c}(\rho,\psi) \models M : v \ \wedge_{i=0}^{j} \ (\rho,\psi) \models M_i : v_i \ \wedge \ (\rho,\kappa) \models P : \psi \cup \psi' \ \wedge \\ \forall \{V_1,\ldots,V_k\}_{V_0}[\psi''] \in v : \ \wedge_{i=0}^{j} V_i \ \mathsf{E} \ v_i \Rightarrow \\ (\wedge_{i=j+1}^{k} \ V_i \in \rho(\lfloor x_i \rfloor) \ \wedge \\ \psi' \subseteq \psi'' \ \cup \ \psi''[[\![V_{j+1}]\!] \mapsto \lfloor x_{j+1} \rfloor, \ldots, [\![V_k]\!] \mapsto \lfloor x_k \rfloor])\end{array}}{(\rho,\kappa) \models \text{ decrypt } M \text{ as } \{M_1,\ldots,M_j; x_{j+1},\ldots,x_k\}_{M_0} \text{ in } P : \psi}$$

# Beviser for den nye Analyse

- Vi har bevist robust sikkerhed for analysen (samt alle hjælpesætninger).
- Vi mangler Conjecture 6.5 (Existence of the Least Solution) samt Conjecture 6.9 (Type Checking is a Subset of CFAC).
- Beviserne er lavet under følgende ændringer:

(CFAC Decrypt)

$$\frac{\begin{array}{c}(\rho,\psi) \models M : v \ \wedge_{i=0}^{j} \ (\rho,\psi) \models M_i : v_i \ \wedge \ (\rho,\kappa) \models P : \psi \cup \psi' \ \wedge \\ \forall \{V_1,\ldots,V_k\}_{V_0}[\psi''] \in v : \ \wedge_{i=0}^{j} V_i \ \mathsf{E} \ v_i \Rightarrow \\ (\wedge_{i=j+1}^{k} \ V_i \in \rho(\lfloor x_i \rfloor) \ \wedge \\ \psi' \subseteq \psi'' \ \cup \ \psi''[[\![V_{j+1}]\!] \mapsto \lfloor x_{j+1} \rfloor, \ldots, [\![V_k]\!] \mapsto \lfloor x_k \rfloor])\end{array}}{(\rho,\kappa) \models \text{ decrypt } M \text{ as } \{M_1,\ldots,M_j; x_{j+1},\ldots,x_k\}_{M_0} \text{ in } P : \psi}$$

- F.eks. fra $\{A\}_K[!\mathrm{begun}(A)]$ overfører vi $!\mathrm{begun}(A)$ og $!\mathrm{begun}(x_1)$.

# Konklusion

# Praktisk Anvendelighed

- Protokollen skal manuelt modelleres i en brugbar proceskalkyle.

# Praktisk Anvendelighed

- Protokollen skal manuelt modelleres i en brugbar proceskalkyle.
- Sikkerhedsegenskaber skal manuelt defineres.

# Praktisk Anvendelighed

- Protokollen skal manuelt modelleres i en brugbar proceskalkyle.
- Sikkerhedsegenskaber skal manuelt defineres.
- Metoderne er ikke komplette.

# Praktisk Anvendelighed

- Protokollen skal manuelt modelleres i en brugbar proceskalkyle.
- Sikkerhedsegenskaber skal manuelt defineres.
- Metoderne er ikke komplette.
- Nogle metoder kræver “hjælp”.

# Praktisk Anvendelighed

- Protokollen skal manuelt modelleres i en brugbar proceskalkyle.
- Sikkerhedsegenskaber skal manuelt defineres.
- Metoderne er ikke komplette.
- Nogle metoder kræver “hjælp”.
- Analysen skal fortolkes.

# Praktisk Anvendelighed

- Protokollen skal manuelt modelleres i en brugbar proceskalkyle.
- Sikkerhedsegenskaber skal manuelt defineres.
- Metoderne er ikke komplette.
- Nogle metoder kræver “hjælp”.
- Analysen skal fortolkes.
- Kun nogle metoder er implementeret (jf. bl.a. LySaTool, Cryptyc og ProVerif).

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:
  - Vi kan da typechecke for dynamisk autenticitet.

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:
  - Vi kan da typechecke for dynamisk autenticitet.
  - Typer kan i nogle tilfælde udledes gennem information fra kontrol-flow-analysen.

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:
  - Vi kan da typechecke for dynamisk autenticitet.
  - Typer kan i nogle tilfælde udledes gennem information fra kontrol-flow-analysen.
- Vi har lavet et kontrol-flow analyse til verificering af korrespondancer:

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:
  - Vi kan da typechecke for dynamisk autenticitet.
  - Typer kan i nogle tilfælde udledes gennem information fra kontrol-flow-analysen.
- Vi har lavet et kontrol-flow analyse til verificering af korrespondancer:
  - Vi formoder den er mindst lige så “stærk” som vores type system.

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:
  - Vi kan da typechecke for dynamisk autenticitet.
  - Typer kan i nogle tilfælde udledes gennem information fra kontrol-flow-analysen.
- Vi har lavet et kontrol-flow analyse til verificering af korrespondancer:
  - Vi formoder den er mindst lige så “stærk” som vores type system.
  - Ny tilgang til verificering af korrespondancer.

# Konklusion

- Vi har lavet et typesystem til verificering af korrespondancer:
  - Kan gøre brug af variabler i korrespondancer.
- Vi har vist at dynamisk autenticitet kan oversættes til korrespondancer:
  - Vi kan da typechecke for dynamisk autenticitet.
  - Typer kan i nogle tilfælde udledes gennem information fra kontrol-flow-analysen.
- Vi har lavet et kontrol-flow analyse til verificering af korrespondancer:
  - Vi formoder den er mindst lige så “stærk” som vores type system.
  - Ny tilgang til verificering af korrespondancer.