Detecting and Preventing Type Flaws:

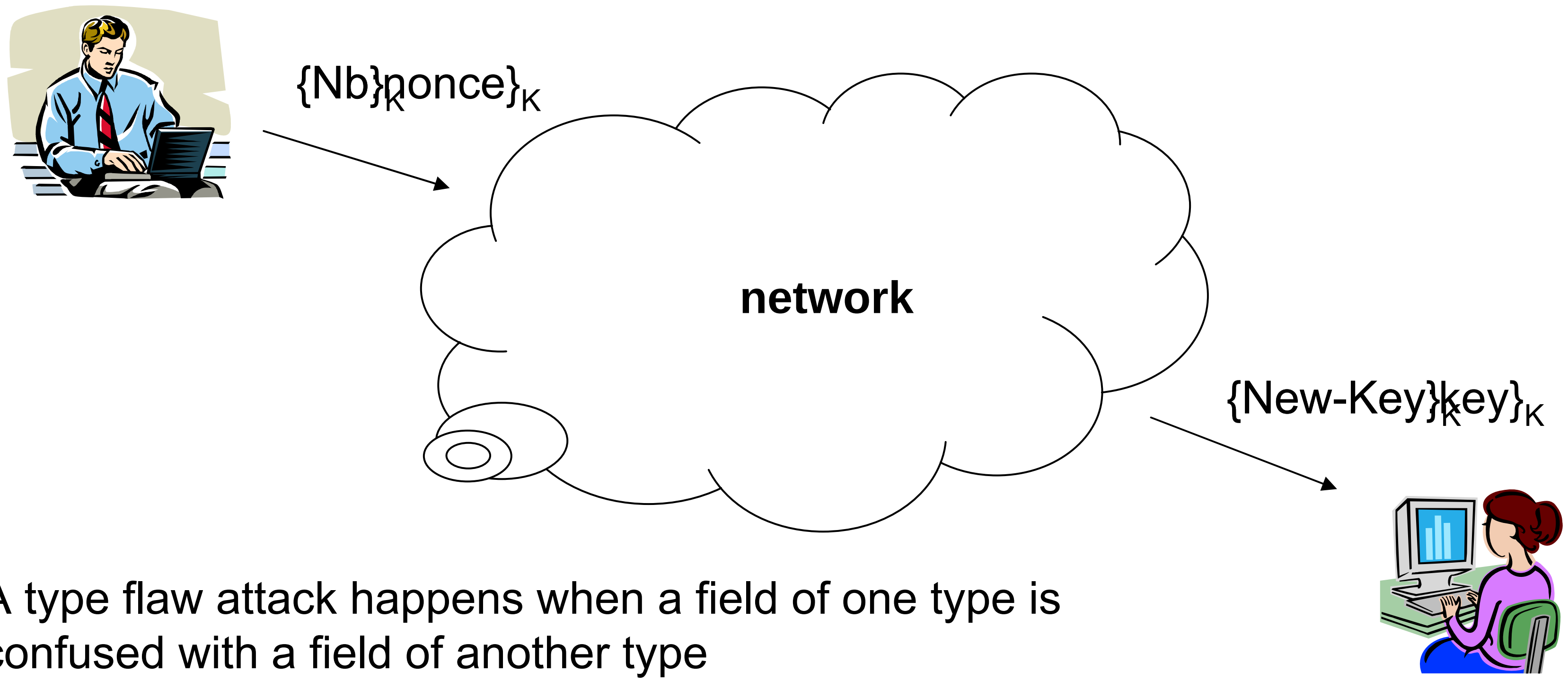
a Control Flow Analysis with tags

Chiara Bodei, Pierpaolo Degano: Pisa University

Han Gao: Technical University of Denmark

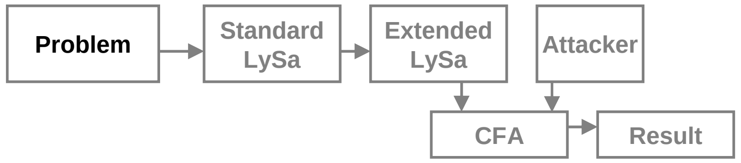
Linda Brodo: Sassari University

What is a Type Flaw Attack?



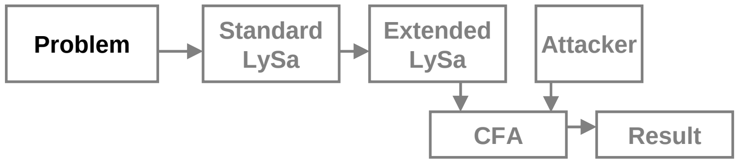
A type flaw attack happens when a field of one type is confused with a field of another type

Why not just use tags? It requires extra computational power and network transmission band

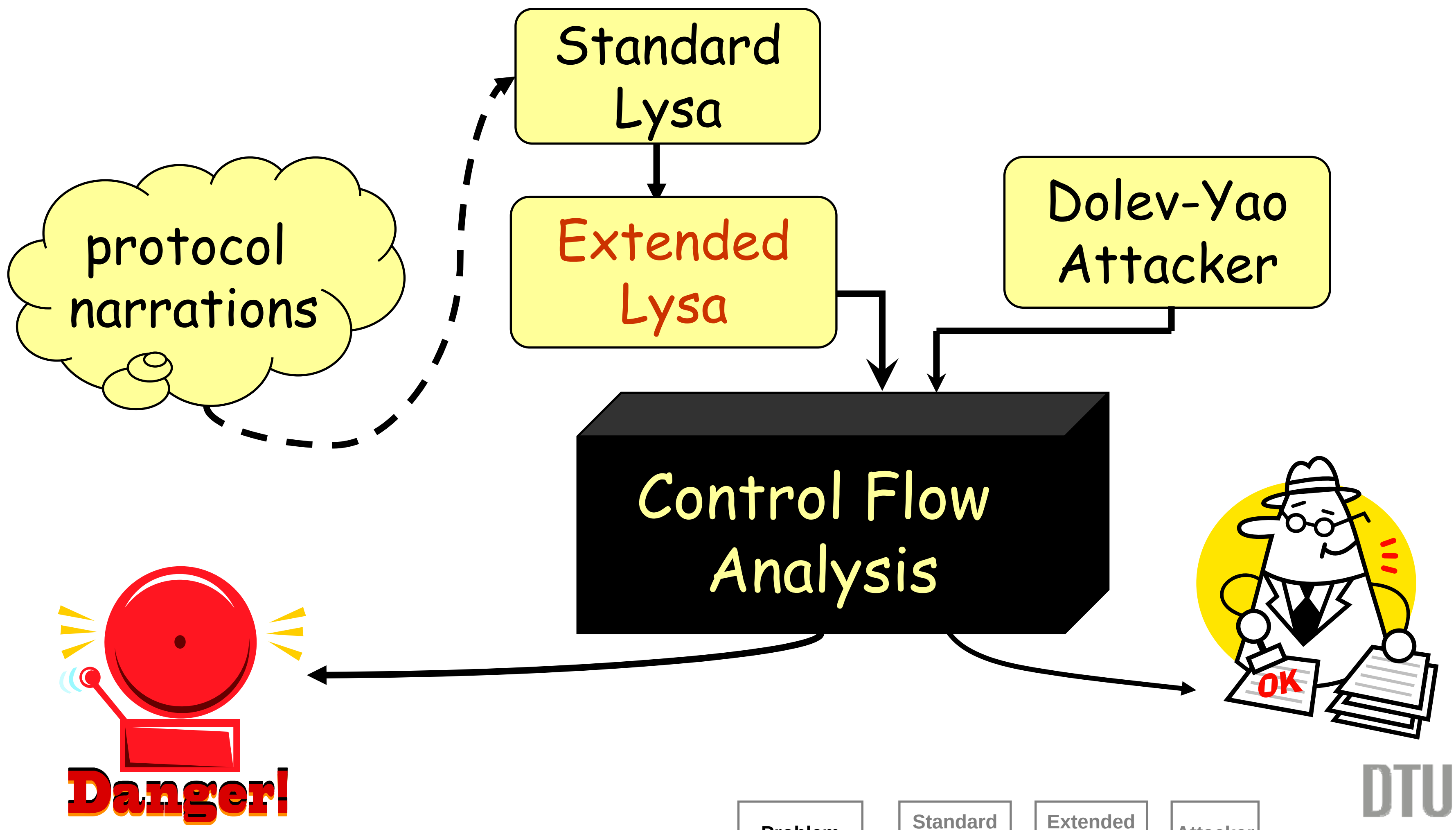


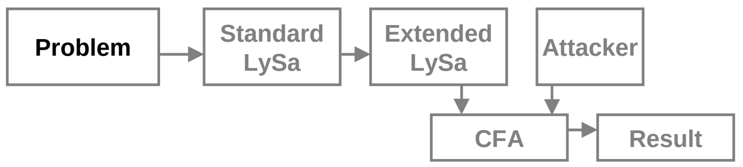
The Goal

- Static Analysis
- Flexibility
  - Detection
    - Define the expected types of fields, check the consistence of types after the protocol execution
  - Prevention
    - Associate tags with fields, abort the protocol execution when type-mismatched



Whole Picture





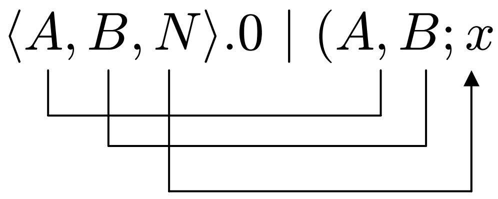
Standard LySa Calculus

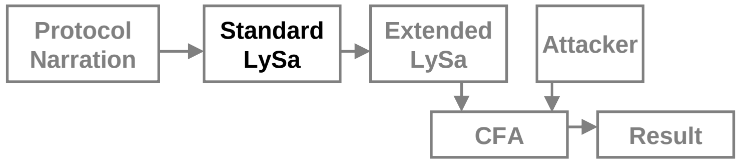
One Global Channel

$$
\begin{aligned}
E &::= n \mid x \mid \{E_1, \ldots, E_k\}_{E_0} \\
P &::= \langle E_1, \ldots, E_k \rangle.P \\
&\mid\quad (E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P \\
&\mid\quad \text{decrypt } E \text{ as } \{E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\}_{E_0} \text{ in } P \\
&\mid\quad (\nu\ n)P \mid P_1|P_2 \mid !P \mid 0
\end{aligned}
$$

Pattern Matching and Variable Binding

$$\langle A, B, N \rangle.0 \mid (A, B; x).\langle x \rangle.0 \rightarrow 0 \mid \langle N \rangle.0$$





Why Extension?

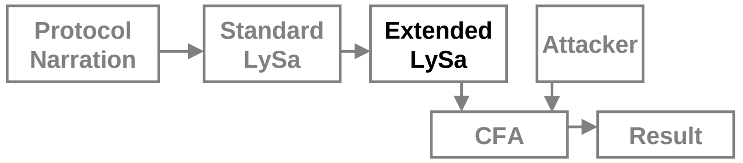
$$\langle A, \boxed{B, N}\rangle.0 \mid (A, \boxed{B; x}).\langle x\rangle.0 \to 0 \mid \langle N\rangle.0$$

- Flexibility in pattern matching and variable binding

$$\langle A, \boxed{N, B}\rangle.0 \mid (A, \boxed{x, B}).\langle x\rangle.0 \to 0 \mid \langle N\rangle.0$$

- Distinguish between the defined occurrences and used occurrences

$$\langle A, N, B\rangle.0 \mid (A, \underset{\text{Defined Occurrence}}{x}; B).\langle \underset{\text{Used Occurrence}}{x}\rangle.0 \to 0 \mid \langle N\rangle.0$$



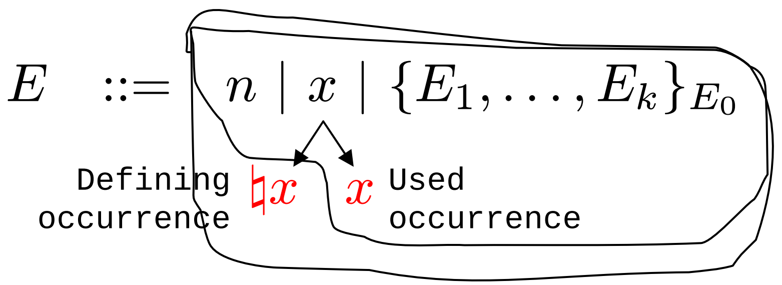
Extended Lysa Calculus

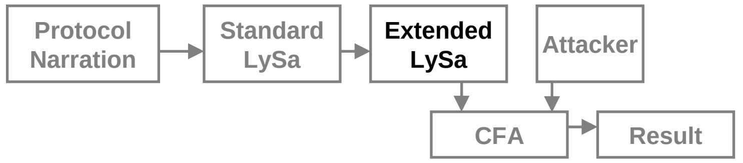
$$Tag \quad ::= \quad agent \mid nonce \mid key \mid \ldots$$

$$\begin{array}{lll} T & ::= & \textit{type terms} \\ & Tag & \text{type tag} \\ & t & \text{(use) type variable} \end{array} \qquad \begin{array}{lll} \mathcal{T} & ::= & \textit{matching type terms} \\ & T & \text{type term} \\ & \sharp t & \text{defining type variable} \end{array}$$

$$\begin{array}{lll} S & ::= & \textit{standard terms} \\ & n & \text{name} \\ & x & \text{(use) variable} \end{array} \qquad \begin{array}{lll} \mathcal{S} & ::= & \textit{matching standard terms} \\ & S & \text{standard term} \\ & \natural x & \text{defining variable} \end{array}$$

$$\begin{array}{lll} E & ::= & \textit{closed terms} \\ & S & \text{standard terms} \\ & T & \text{type terms} \\ & \{E_1, \cdots, E_k\}_{E_0} & \text{encryption} \end{array} \qquad \begin{array}{lll} M & ::= & \textit{matching terms} \\ & \mathcal{S} & \text{matching standard term} \\ & \mathcal{T} & \text{matching type term} \\ & \{M_1, \ldots, M_k\}_{E_0} & \text{matching encryption} \end{array}$$


$E \quad ::= \quad n \mid x \mid \{E_1, \ldots, E_k\}_{E_0}$
Defining occurrence
$\natural x$
x
Used occurrence



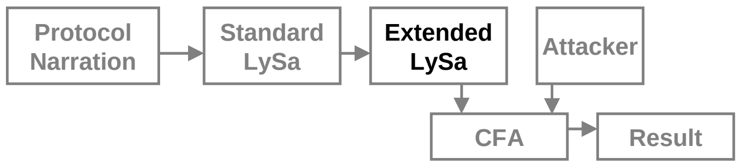
Extended Lysa Calculus

$$
\begin{array}{lll}
P & ::= & \langle E_1, \ldots, E_k \rangle.P \\
 & \mid & (M_1, \ldots, M_k).P \\
 & \mid & decrypt\ E\ as\ \{M_1, \ldots, M_k\}^{l}_{E_0}\ in\ P \\
 & \mid & (\nu\ \sharp t : Tag)P \\
 & \mid & (\nu\ n)P \mid P_1 \mid P_2 \mid !P \mid 0
\end{array}
$$

Expected value

Extended Pattern Matching and Variable Binding

$$\langle N, nonce\rangle.0 \mid (\nu\ \sharp t : nonce)(\natural x, \sharp t).\langle x, t\rangle.0 \rightarrow 0 \mid \langle N, nonce\rangle.0$$



Extended Pattern Matching

$$\begin{aligned} P = \mathit{decrypt} \quad & \{A, t\}_K \ \mathit{as} \\ & \{\natural x, \mathit{nonce}\}_K^{l_P} \ \mathit{in} \ P' \end{aligned}$$

Succeeds when $t = \mathit{nonce}$

$$\begin{aligned} Q = \mathit{decrypt} \quad & \{A, \mathit{nonce}\}_K \ \mathit{as} \\ & \{\natural x, \sharp t\}_K^{l_Q} \ \mathit{in} \ Q \end{aligned}$$

Always succeeds

$$\begin{tikzcd}
\text{Protocol Narration} \arrow[r] & \text{Standard LySa} \arrow[r] & \textbf{Extended LySa} \arrow[d] & \text{Attacker} \arrow[dl] \\
& & \text{CFA} \arrow[r] & \text{Result}
\end{tikzcd}$$

The Control Flow Analysis

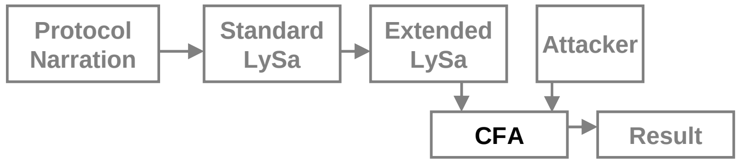
- Over-approximate the protocol behaviour
- The values of the variables and type variables $\rho : X \cup T \rightarrow \mathcal{P}(Val)$
- The messages flowing on the network

$$\kappa \subseteq \mathcal{P}(Val^*)$$

- For example:

$$\langle N, nonce \rangle \in \kappa$$

$$N \in \rho(x)$$



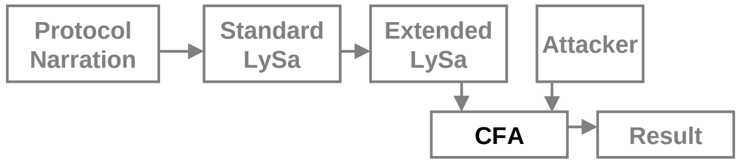
The Error and Type Components

- The error component ψ collects labels of decryption where type-mismatching may happen. For example,

$$l \in \psi$$

- The type component Γ collects the declared value of each defining type variable. For example,

$$(\nu \sharp t : key)P \Rightarrow (\sharp t, key) \in \Gamma$$



The Analysis

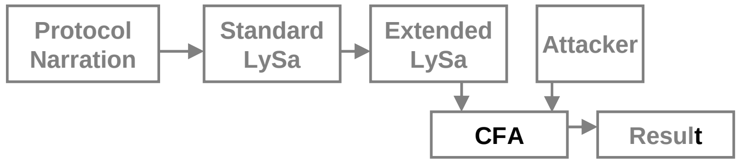
The analysis is specified as the judgement

$$\rho, \kappa, \Gamma \models P : \psi$$

and auxiliary judgement for terms

$$\rho \models E : \vartheta$$

where $\vartheta \subseteq \mathcal{P}(Val)$ is the values that E may evaluate to

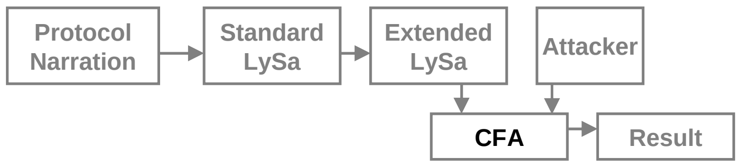


Judgement for Decryption

- At each decryption point, check whether each defined variable has the expected type

$$
\begin{array}{ll}
\dfrac{
\begin{array}{l}
\rho \models E : \vartheta \ \wedge \\
\rho \models E_0 : \vartheta_0 \ \wedge \\
\forall \{v_1, v_2\}_{v_0} \in \vartheta : v_0 \in \vartheta_0 \Rightarrow \\
\quad match(v_1, M_1) \wedge match(v_2, M_2) \Rightarrow \\
\quad\quad bind(v_1, M_1) \wedge bind(v_2, M_2) \ \wedge \\
\quad\quad\quad {\color{blue} chk(v_1, M_1, \Gamma, l) \wedge chk(v_2, M_2, \Gamma, l)} \ \wedge \\
\quad\quad\quad\quad \rho, \kappa, \Gamma \models P : \psi
\end{array}
}{
\rho, \kappa, \Gamma \models decrypt\ E\ as\ \{M_1, M_2\}^{l}_{E_0}\ in\ P : \psi
}
&
\begin{array}{l}
\text{evaluate term} \\
\text{evaluate key} \\
\text{for all encrypted values} \\
\text{pattern matching} \\
\text{variable binding} \\
\text{type checking} \\
\text{analyse the rest}
\end{array}
\end{array}
$$

$match(v, M)$:	$M\ is\ S\ or\ T$
$bind(v, M)$:	$M\ is\ \natural x\ or\ \sharp t$
$chk(v, M, \Gamma, l)$:	$M\ is\ \sharp t$

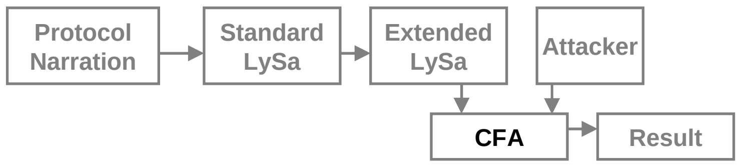


The Control Flow Analysis

- At each decryption point, check whether each defined variable has the expected type

$$\begin{tikzcd}
& \{N, \arrow[d, leftrightarrow] & nonce\}_K \in \rho(y) \arrow[d, leftrightarrow] \\
(\nu\ \sharp t : nonce)\ decrypt\ y\ as & \{\natural x, & \sharp t\}^l_K\ in\ P
\end{tikzcd}$$

$$\begin{tikzcd}
(\nu\ \sharp t : key)\ decrypt\ y\ as & \{\natural x, \arrow[d, leftrightarrow] & \sharp t\}^l_K\ in\ P \arrow[d, leftrightarrow] \\
& \{N, & nonce\}_K \in \rho(y)
\end{tikzcd}$$



Attacker

$$P_\bullet \mid P_{sys}$$

- Learn knowledge
  - By eavesdrop
  - By decryption
- Generate knowledge
  - Generate new names
  - Generate new encryptions
- Send out messages
  - Not able to touch the type of each field

$$\begin{tikzcd}
\text{Protocol Narration} \arrow[r] & \text{Standard LySa} \arrow[r] & \text{Extended LySa} \arrow[d] & \textbf{Attacker} \arrow[dl] \\
 & & \text{CFA} \arrow[r] & \text{Result}
\end{tikzcd}$$

Example

$$A \rightarrow \quad : \{Na\}_K$$

$$\rightarrow B : \{K'\}_K$$

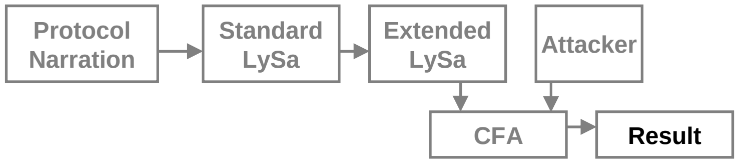
GOAL: **Detect** type flaw attacks

$$\langle\{N, nonce\}_K\rangle.0$$

$$\mid \quad (\nu \sharp t : key) \quad (\natural x_{enc}).$$
$$decrypt\ x_{enc}\ as\ \{\natural x, \sharp t\}^l_K\ in\ 0$$

Expected type is *key*

Variable binding is allowed



Example Cont.

$$A \rightarrow \quad : \{Na\}_K$$

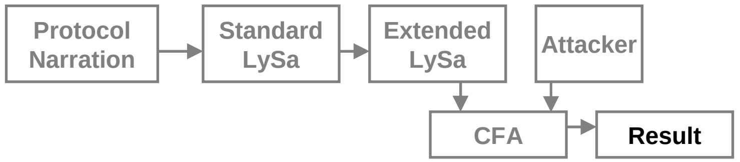
$$\rightarrow B : \{K'\}_K$$

GOAL: **Prevent** type flaw attacks

$$\langle\{N, nonce\}_K\rangle.0$$

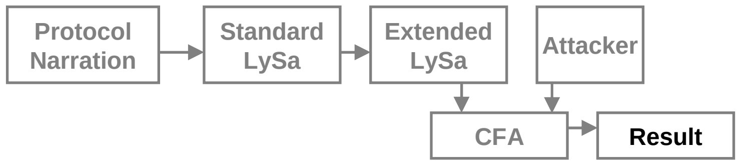
$$\mid \quad (\natural x_{enc}).decrypt\ x_{enc}\ as\ \{\natural x, \underset{\uparrow}{key}\}^l_K\ in\ 0$$

x has to be of type key



Conclusion

- Type Flaw Attacks
- Control Flow Analysis
  - Both prescriptive and descriptive
- A Number of Experiments
  - Woo and Lam Protocol π_1
  - Andrew Secure RPC Protocol
- Current Work
  - Complex Type Flaw Attacks: a field is confused with a concatenation of fields



Thanks!

Question?