Usuba

Samuel Vivien, sous l'encadrement de Pierre-Évariste Dagand

31 août 2023

1. USUBA aujourd'hui
2. Un système de type pour USUBA
3. 4 sémantiques pour USUBA
4. Conclusion

1 USUBA aujourd'hui

2 Un système de type pour USUBA

3 4 sémantiques pour USUBA

4 Conclusion

```
table SubColumn (input:v4) returns (out:v4)
    { 6, 5, 12, 10, 1, 14, 7, 9, 11, 0, 3, 13, 8, 15, 4, 2 }

table SubColumn (input:v4) returns (out:v4)
    { 6, 5, 12, 10, 1, 14, 7, 9, 11, 0, 3, 13, 8, 15, 4, 2 }

node ShiftRows (input:u16[4]) returns (out:u16[4])
vars
let
    out[0] = input[0];          out[1] = input[1] <<< 1;
    out[2] = input[2] <<< 12; out[3] = input[3] <<< 13
tel

node Rectangle (plain:u16[4],key:u16[26][4])
returns (cipher:u16[4])
vars tmp : u16[26][4]
let
    tmp[0] = plain;
    forall i in [0,24] {
        tmp[i+1] = ShiftRows( SubColumn( tmp[i] ^ key[i] ) )
    }
    cipher = tmp[25] ^ key[25]
tel
```

Spécificités du langage

- Pas de conditionnelles (`if` ou `while`)

Spécificités du langage

- Pas de conditionnelles (`if` ou `while`)
- Pas d'accès mémoire dynamiques

Spécificités du langage

- Pas de conditionnelles (`if` ou `while`)
- Pas d'accès mémoire dynamiques

Cela permet d'avoir un temps d'exécution indépendant des valeurs du calcul

Limitations du langage

- Pas de système de type

Limitations du langage

- Pas de système de type
- Pas de spécification de la sémantique

Limitations du langage

- Pas de système de type
- Pas de spécification de la sémantique
- La sémantique implémenté est non compositionnelle !

Limitations du langage

- Pas de système de type
- Pas de spécification de la sémantique
- La sémantique implémenté est non compositionnelle !

On prend $v = [[0, 1], [2, 3], [4, 5]]$

Limitations du langage

- Pas de système de type
- Pas de spécification de la sémantique
- La sémantique implémenté est non compositionnelle !

On prend $v = [[0, 1], [2, 3], [4, 5]]$
Donc $v[0, 1][1] = [1, 3]$

Limitations du langage

- Pas de système de type
- Pas de spécification de la sémantique
- La sémantique implémenté est non compositionnelle !

On prend $v = [[0, 1], [2, 3], [4, 5]]$
Donc $v[0, 1][1] = [1, 3]$
On prend $\{x = v[0, 1];\ y = x[1]\}$, on obtient donc $y = [2, 3]$

Limitations du langage

- Pas de système de type
- Pas de spécification de la sémantique
- La sémantique implémenté est non compositionnelle !

Correction : distinguer $v[0,1;\ 1]$ de $v[0,1][1]$

Amélioration des appels

```
node MapRectangle (plain:u16[64][4], key:u16[64][26][4])
returns (cipher:u16[64][4])
vars
let
    forall i in [0,63] {
        chiper[i] = Rectangle( plain, key )
    }
tel
```

Amélioration des appels

```
node MapRectangle (plain:u16[64][4],key:u16[64][26][4])
returns (cipher:u16[64][4])
vars
let
    forall i in [0,63] {
        chiper[i] = Rectangle( plain[i], key[i] )
    }
tel

node MapRectangle (plain:u16[64][4],key:u16[64][26][4])
returns (cipher:u16[64][4])
vars
let
    chiper = Rectangle[64]( plain[i], key[i] )
tel
```

1 USUBA aujourd'hui

2 Un système de type pour USUBA

3 4 sémantiques pour USUBA

4 Conclusion

Grammaire des types

$$
\begin{array}{l}
\mathit{dir} ::= \\
\quad \mid \mathbf{V} \\
\quad \mid \mathbf{H} \\
\quad \mid d \\
\\
\mathit{size} ::= \\
\quad \mid s \\
\quad \mid z \\
\\
\sigma ::= \\
\quad \mid \mathbf{U}\ \mathit{dir}\ \mathit{size}
\end{array}
$$

Grammaire des types

$$
\begin{array}{l}
\mathit{dir} ::= \\
\quad \mid \mathbf{V} \\
\quad \mid \mathbf{H} \\
\quad \mid d \\
\\
\mathit{size} ::= \\
\quad \mid s \\
\quad \mid z \\
\\
\sigma ::= \\
\quad \mid \mathbf{U}\ \mathit{dir}\ \mathit{size}
\end{array}
$$

U V 32 : entier 32 bits classique.
U H 64 : entier 64 bits découpé dans 64 registres.
$u32 = \mathbf{U}\ \mathit{dir}\ 32$ et $v4 = \mathbf{U}\ \mathit{dir}\ \mathit{size}[4]$.

Grammaire des types

$$
\begin{array}{l}
dir ::= \\
\quad \mid \mathbf{V} \\
\quad \mid \mathbf{H} \\
\quad \mid d
\end{array}
$$

$$
\begin{array}{l}
size ::= \\
\quad \mid s \\
\quad \mid z
\end{array}
$$

$$
\begin{array}{l}
\sigma ::= \\
\quad \mid \mathbf{U}\ dir\ size
\end{array}
$$

$$
\begin{array}{l}
\tau ::= \\
\quad \mid \sigma \\
\quad \mid \tau[a]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{T} ::= \\
\quad \mid \overline{\tau_n}
\end{array}
$$

Grammaire des types

$$
\begin{aligned}
dir ::= & \\
& \mid \mathbf{V} \\
& \mid \mathbf{H} \\
& \mid d
\end{aligned}
$$

$$
\begin{aligned}
size ::= & \\
& \mid s \\
& \mid z
\end{aligned}
$$

$$
\begin{aligned}
\sigma ::= & \\
& \mid \mathbf{U}\ dir\ size
\end{aligned}
$$

$$
\begin{aligned}
\tau ::= & \\
& \mid \sigma \\
& \mid \tau[a]
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{T} ::= & \\
& \mid \overline{\tau_n}
\end{aligned}
$$

$$
\begin{aligned}
typc ::= & \\
& \mid \mathbf{Arith}\ \tau \\
& \mid \mathbf{Logic}\ \tau \\
& \mid \mathbf{Shift}\ \tau\ a_2
\end{aligned}
$$

$$
\begin{aligned}
A ::= & \\
& \mid \overline{typc_n}
\end{aligned}
$$

Règles de typage

$$\frac{\Gamma, P, A \vdash_E e_1 : \tau \quad \Gamma, P, A \vdash_E e_2 : \tau \quad A \vdash \textbf{ClassOf}\ \mathit{binop}\ \tau}{\Gamma, P, A \vdash_E e_1\ \mathit{binop}_\tau\ e_2 : \tau}\ \textsc{Binop}$$

$$\frac{\Gamma, P, A \vdash_E \overline{e_n : \mathcal{T}_n}}{\Gamma, P, A \vdash_E (\overline{e_n}) : \overline{\mathcal{T}_n}}\ \textsc{Tuple}$$

Règles de typage des appels de nœuds

$$\frac{\begin{array}{c} P \vdash f : \forall\, \overline{d_n}\,, \forall\, \overline{s_m}\,, \overline{typc_j} \Rightarrow \mathcal{T}_1 \to \mathcal{T}_2 \\ \Gamma, P, A \vdash_E (\,\overline{e_n}\,) : \mathcal{T}_1' \\ A \vdash typc_j[\,\overline{d_n \leftarrow d_n'}\;;\;\overline{s_m \leftarrow s_m'}\,] \\ \mathcal{T}_1[\,\overline{d_n \leftarrow d_n'}\;;\;\overline{s_m \leftarrow s_m'}\,] = \sigma_x\,\overline{[\ell_g]}\,\overline{[z_q]} \\ \mathcal{T}_2[\,\overline{d_n \leftarrow d_n'}\;;\;\overline{s_m \leftarrow s_m'}\,] = \sigma_y'\,\overline{[\ell_g]}\,\overline{[z_r']} \\ \mathcal{T}_1' \cong \sigma_x\,\overline{[\ell_g]}\,\overline{[\ell_h']}\,\overline{[z_q]} \end{array}}{\Gamma, P, A \vdash_E [\,\overline{\ell_g}\,] f [\,\overline{\ell_h'}\,](\,\overline{e_n}\,) : \sigma_y'\,\overline{[\ell_g]}\,\overline{[\ell_h']}\,\overline{[z_r']}}\ \textsc{Fun}$$

Deux nouvelles constructions

- Les coercions explicites

Deux nouvelles constructions

- Les coercions explicites
- Les constructeurs de tableaux (pour typer $x + (x[0], x[1])$)

1 USUBA aujourd'hui

2 Un système de type pour USUBA

3 4 sémantiques pour USUBA

4 Conclusion

Sémantique par évaluation

1. On évalue tout dans l'ordre

Sémantique par évaluation

1. On évalue tout dans l'ordre
2. Permet de gérer des équations de modifications

Sémantique par évaluation

1. On évalue tout dans l'ordre
2. Permet de gérer des équations de modifications
3. Sémantique la plus proche de celle implémenté

Sémantique relationnelle

1. On définie une propriété $e \mapsto v$.

Sémantique relationnelle

1. On définie une propriété $e \mapsto v$.
2. Sémantique indépendante de l'ordre des équations

Sémantique relationnelle

1. On définie une propriété $e \mapsto v$.
2. Sémantique indépendante de l'ordre des équations
3. Sémantique non calculatoire

Sémantique relationnelle

1. On définie une propriété $e \mapsto v$.
2. Sémantique indépendante de l'ordre des équations
3. Sémantique non calculatoire
4. Accepte beaucoup de systèmes comme $\{y = y\}$

Sémantique par tri topologique

1. Remonte l'ordre des évuations pour calculer les valeurs

Sémantique par tri topologique

1. Remonte l'ordre des évuations pour calculer les valeurs
2. Peu maniable pour de la preuve de préservation de la sémantique

Sémantique par point fixe

1. Essaye de calculer les équations par passages successifs sur le système

Sémantique par point fixe

1. Essaye de calculer les équations par passages successifs sur le système
2. Sous ensemble stricte de la sémantique relationnelle

1 USUBA aujourd'hui

2 Un système de type pour USUBA

3 4 sémantiques pour USUBA

4 Conclusion

Conclusion

Merci de m'avoir écouté