

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[i], key[i] )  
    }  
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] {
    cipher[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
  cipher = 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

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

Grammaire des types

$$\begin{array}{l} \textit{dir} ::= \\ | \textbf{V} \\ | \textbf{H} \\ | d \end{array}$$
$$\begin{array}{l} \textit{size} ::= \\ | s \\ | z \end{array}$$
$$\begin{array}{l} \sigma ::= \\ | \textbf{U} \textit{ dir size} \end{array}$$

U V 32 : entier 32 bits classique.

U H 64 : entier 64 bits découpé dans 64 registres.

$u32 = \textbf{U} \textit{ dir } 32$ et $v4 = \textbf{U} \textit{ dir size}[4]$.

Grammaire des types

$$\begin{array}{l} \textit{dir} ::= \\ | \mathbf{V} \\ | \mathbf{H} \\ | d \end{array}$$
$$\begin{array}{l} \textit{size} ::= \\ | s \\ | z \end{array}$$
$$\begin{array}{l} \sigma ::= \\ | \mathbf{U} \textit{ dir size} \end{array}$$
$$\begin{array}{l} \tau ::= \\ | \sigma \\ | \tau[a] \end{array}$$
$$\begin{array}{l} \mathcal{T} ::= \\ | \overline{\tau_n} \end{array}$$

Grammaire des types

$$\text{dir} ::=$$

V
H
d

$$\text{size} ::=$$

s
z

$$\sigma ::=$$

U dir size

$$\tau ::=$$

σ
$\tau[a]$

$$\mathcal{T} ::=$$

$\overline{\tau_n}$

$$\text{typc} ::=$$

Arith τ
Logic τ
Shift τa_2

$$A ::=$$

$\overline{\text{typc}_n}$

Règles de typage

$$\frac{\begin{array}{l} \Gamma, P, A \vdash_E e_1 : \tau \\ \Gamma, P, A \vdash_E e_2 : \tau \\ A \vdash \mathbf{ClassOf} \, binop \, \tau \end{array}}{\Gamma, P, A \vdash_E e_1 \, binop_{\tau} \, e_2 : \tau} \quad \text{BINOP}$$

$$\frac{\Gamma, P, A \vdash_E \overline{e_n} : \overline{\mathcal{T}_n}}{\Gamma, P, A \vdash_E (\overline{e_n}) : \overline{\mathcal{T}_n}} \quad \text{TUPLE}$$

Règles de typage des appels de nœuds

$$\begin{array}{c}
 P \vdash f : \forall \overline{d_n}, \forall \overline{s_m}, \overline{\text{typc}_j} \Rightarrow \mathcal{T}_1 \rightarrow \mathcal{T}_2 \\
 \Gamma, P, A \vdash_E (\overline{e_n}) : \mathcal{T}'_1 \\
 \hline
 A \vdash \overline{\text{typc}_j}[\overline{d_n \leftarrow d'_n} ; \overline{s_m \leftarrow s'_m}] \\
 \hline
 \mathcal{T}_1[\overline{d_n \leftarrow d'_n} ; \overline{s_m \leftarrow s'_m}] = \overline{\sigma_x[\ell_g][z_q]} \\
 \mathcal{T}_2[\overline{d_n \leftarrow d'_n} ; \overline{s_m \leftarrow s'_m}] = \overline{\sigma'_y[\ell_g][z'_r]} \\
 \mathcal{T}'_1 \cong \overline{\sigma_x[\ell_g][\ell'_h][z_q]} \\
 \hline
 \Gamma, P, A \vdash_E [\overline{\ell_g}]f[\overline{\ell'_h}](\overline{e_n}) : \overline{\sigma'_y[\ell_g][\ell'_h][z'_r]}
 \end{array}
 \quad \text{FUN}$$

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 définit une propriété $e \mapsto v$.

Sémantique relationnelle

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

Sémantique relationnelle

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

Sémantique relationnelle

- ① On définit 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 évaluations pour calculer les valeurs

Sémantique par tri topologique

- ① Remonte l'ordre des évaluations 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 Essayer 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