

L3-MI / PROG

TP#4 : Fonction de hachage polynomiale

2022-W40

Instructions de rendu

Ce TP fera possiblement l'objet d'un rendu commun avec avec un futur TP sur les tables de hachage.

Présentation

Le but de cet exercice est d'implémenter une famille de fonction de hachage dite *polynomiale*. Une fonction de hachage est une application $H : \{0, 1\}^* \rightarrow \mathcal{I}$, où \mathcal{I} est fixé par le contexte, qui transforme un « message » de taille variable potentiellement importante en un « haché » (ou « empreinte », ou « condensat ») de taille fixée, généralement courte.

Une propriété importante d'une famille de fonction de hachage \mathcal{H} est sa capacité à être « universelle ». Informellement, on souhaite que $\Pr_{H \in \mathcal{H}}[H(x) = H(x' \neq x)] \leq \varepsilon$, où la probabilité est calculée sur le choix uniforme de H dans la famille de fonction \mathcal{H} , et où ε est « petit ».

Le principe d'une famille de fonction de hachage polynomiale est le suivant :

1. on commence par fixer un corps fini \mathbb{F}_q (qui dans notre cas sera $\mathbb{F}_{2^{33}-9} \cong \mathbb{Z}/2^{33}-9\mathbb{Z}$, le corps des entiers modulo le nombre premier $2^{33}-9 = 8589934583$);
2. on choisit un paramètre $k \in \mathbb{F}_q \setminus \{0\}$ qui définit une instance de la famille de fonction. C'est à dire que $\mathcal{H} = \{H_k, k \in \mathbb{F}_q \setminus \{0\}\}$;
3. on découpe l'entrée m de la fonction de hachage en l blocs m_1, \dots, m_l , chacun étant vu comme la représentation binaire d'un nombre se trouvant dans l'intervalle $\llbracket 0, s \rrbracket$, $s < q$;
4. la sortie de la fonction de hachage $H_k(m)$ est le résultat de l'évaluation du polynôme $M := \sum_{i=1}^l m_i X^i \in \mathbb{F}_{2^{33}-9}[X]$ (ou alternativement $\sum_{i=1}^l m_i X^{l+1-i}$) en k .

Les bonnes propriétés d'universalité d'une telle famille de fonction viennent du fait que si m et m' sont des messages découpés en au plus l blocs, $\Pr_{k \in \mathbb{F}_q}[H_k(m) = H_k(m')] = \Pr_{k \in \mathbb{F}_q}[\text{eval}(M - M', k) = 0] \leq l/q$, puisqu'un polynôme de degré l a au plus l racines distinctes.

Q.1 : Pour implémenter une fonction de hachage polynomiale sur $\mathbb{F}_{2^{33}-9}$, il faut tout d'abord implémenter la multiplication de deux éléments de ce corps, c'est à dire la multiplication de deux entiers modulo $2^{33}-9$.

Écrivez une fonction :

```
// In: 0 <= a, b < 8589934583
// Out: 0 <= x < 8589934583 t.q. x ~ a * b [8589934583]
uint64_t mul339(uint64_t a, uint64_t b);
```

qui implémente cette multiplication. Dans un souci d'efficacité, votre fonction ne pourra utiliser que des divisions (via `/` ou `%`) par des puissances de deux.

Remarques & conseils

- $2^{33} - 9 = 8589934583ULL$
- $2^{32} \in \llbracket 0, 2^{33} - 10 \rrbracket$, mais le produit $2^{32} \times 2^{32} = 2^{64}$ provoque un *overflow* s'il est calculé avec une multiplication standard entre nombres de 64 bits. Une possibilité est donc de représenter les arguments a et b sous la forme $a_1 2^r + a_0$, $b_1 2^r + b_0$, avec un r bien choisi.
- On remarque que $2^{33} \equiv 9 \pmod{2^{33} - 9}$, $2^{34} \equiv 18 \pmod{2^{33} - 9}$, etc. Comment pouvez vous utiliser ceci pour efficacement calculer un nombre $x < 2^{54}$ t.q. $x \equiv (a_1 2^r + a_0) \times (b_1 2^r + b_0) \pmod{2^{33} - 9}$?
- En utilisant la même remarque que précédemment, soit $x \in \llbracket 0, 2^{54} - 1 \rrbracket$, comment pouvez vous calculer $y \in \llbracket 0, 2 \times (2^{33} - 9) - 1 \rrbracket$ et de là $y' \in \llbracket 0, 2^{33} - 10 \rrbracket$, t.q. x , y et y' sont congrus entre eux modulo $2^{33} - 9$? *Indice* : pensez à exploiter la division euclidienne de x par 2^{33} .
- Pensez à borner la taille des quantités que vous manipulez dans votre implémentation, afin de justifier sa correction.

N.B. L'algorithme suggéré est un cas particulier de la réduction dite « de Barrett ».

Testez votre fonction en la comparant sur quelques exemples avec les résultats obtenus par un calcul direct en Python.

Q.2 : Écrivez une fonction :

```
uint64_t hash339(uint64_t k, size_t buflen, uint8_t buf[buflen]);
```

qui implémente une fonction de hachage polynomiale sur $\mathbb{F}_{2^{33}-9}$, où le paramètre k est pris dans $\llbracket 0, 2^{33} - 10 \rrbracket$, et le message de taille `buflen` est stocké dans `buf`.

- On conseille de d'abord écrire une fonction pour les messages dont la taille (en octets) est un multiple de 4, puis de traiter séparément un potentiel dernier bloc incomplet.
- Une implémentation efficace cherchera à évaluer $\sum_{i=1}^l m_i X^{l+1-i}$ par la « méthode de Horner », qui remarque que ce polynôme peut s'écrire $m_l X + X \times (m_{l-1} X + X \times (\dots X \times (m_1 X) \dots))$

Testez votre fonction en la comparant sur quelques exemples avec les résultats obtenus par un calcul direct en Python.