Abstract

Example of mixing text and math

Криптосистема RSA

Криптосистема RSA является, наверное, самой популярной криптографической схемой. Пусть n – составной модуль и e – открытая экспонента. То есть пара (n, e) является открытым ключом криптосистемы. Далее, пусть $m \in Z_n$ – открытый текст. Функция шифрования $f((n, e), m) = m^e \ mod \ n$ гомоморфна относительно умножения открытых текстов. В самом деле, для любых открытых текстов m_1, m_2 и любого открытого ключа k криптограмма произведения равна произведению криптограмм сомножителей: $f(k, m_1 \cdot m_2) = f(k, m_1) \cdot f(k, m_2)$

Криптосистема Эль-Гамаля

Пусть G – циклическая группа порядка p и g – ее порождающий элемент. В качестве секретного ключа криптосистемы выбирается случайный элемент x группы Z_{p-1}. Соответствующий открытый ключ вычисляется по формуле $y = g^x$. Криптограмма открытого текста $m \in G$ вычисляется с помощью функции шифрования $f(y, m) = (y^r \cdot m, g^r)$, где r – случайный элемент группы Z_{p-1}, то есть число r выбирается всякий раз заново, независимо и равновероятно.

Дешифрование криптограммы c_1, c_2 выполняется следующим образом. Сначала вычисляется $c_2^x = g^{r \cdot x}$, откуда $m = \frac{c_1}{c_2^x}$ Функция шифрования криптосистемы Эль-Гамаля гомоморфна относительно операции умножения открытых текстов: криптограмма произведения может быть вычислена как произведение криптограмм сомножителей. Если $f(y, m_1) = (y^{r_1} \cdot m_1, g^{r_1})$ и $f(y, m_2) = (y^{r_2} \cdot m_2, g^{r_2})$, то $f(y, m_1 \cdot m_2)$ можно получить в виде $(y^{r_1} \cdot y^{r_2} \cdot m_1 \cdot m_2, \ g^{r_1} \cdot g^{r_2})$

Example of theorem

Theorem

$$id : \oplus \to_F \oplus$$
$$f : \oplus \to_F \otimes \quad \text{and} \quad g : \otimes \to_F \odot \quad \Rightarrow \quad g \circ f : \oplus \to_F \odot$$

Proof

$$
\begin{array}{lrcl}
id : \oplus \to_F \oplus & \equiv & id \circ \oplus & = & \oplus \circ F\ id \\
& \equiv & \oplus \circ id & = & \oplus \circ F\ id \\
& \equiv & id & = & F\ id
\end{array}
$$

$$
\begin{array}{lrcl}
g \circ f : \oplus \to_F \odot & \equiv & g \circ f \circ \oplus & = & \odot \circ F\ (g \circ f) \\
& \equiv & g \circ \otimes \circ F\ f & = & \odot \circ F\ (g \circ f) \\
& \equiv & \odot \circ F\ g \circ F\ f & = & \odot \circ F\ (g \circ f) \\
& \equiv & F\ g \circ F\ f & = & F\ (g \circ f)
\end{array}
$$

Example of code

Kleisli and Monad type classes connection

```haskell
module Kleisli where

import Prelude hiding ((>>), (=<<), id)

class Category ar where
    id  :: ar a a
    (>>) :: ar a b -> ar b c -> ar a c

class Kleisli m where
    idK  :: a -> m a
    (*>) :: (a -> m b) -> (b -> m c) -> a -> m c

instance Category (->) where
    id x = x
    (>>) = flip (.)

f +> g = f *> (g >> idK)

f =<< x = const x *> f $ ()

f >=> g = \x -> f x >>= g
```

Multidimentional array map

```c
int arrMap(Arr *ar, Arr *ar2, int (*fn)(Arr *, Arr *, ArrAc *)) {
    /* we don't check the second argument since this function can be used for
     * in-place update
     */
    if (!ar) {
        ELOG("null pointer");
        return -1;
    }
    if (!fn) {
        ELOG("null pointer");
        return -1;
    }
    ArrAc ac = { ar->dim, {}, NULL };
    int k = ar->dim - 1;
    while (1) {
        if (fn(ar, ar2, &ac)) {
            ELOG("function app failed");
            return -1;
        }
```

```
        while (k >= 0 && ++ac.ix[k] == ar->size[k])
            ac.ix[k--] = 0;
        if (k >= 0)
            k = ar->dim - 1;
        else
            break;
    }
    return 0;
}
```