McEliece crypto-system

A reference implementation

Bhaskar Biswas and Nicolas Sendrier

Projet SECRET, INRIA, Rocquencourt

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 The crypto-system **5**

3.1 Introduction 5

3.2 Security 6

3.3 Binary Goppa codes 7

3.4 Specification 9

4 Data Structure Documentation **13**

4.1 buff Struct Reference 13

4.2 code_arith Struct Reference 15

4.3 distrib_t Struct Reference 17

4.4 elt Struct Reference 18

4.5 leaf_info_t Struct Reference 20

4.6 lnode Struct Reference 21

4.7 matrix Struct Reference 22

4.8 polynome Struct Reference 24

4.9 precomp Struct Reference 25

4.10 tnode Struct Reference 27

5 File Documentation **29**

5.1 arith.c File Reference 29

5.2 arith.h File Reference 32

5.3 buff.c File Reference 35

5.4 buff.h File Reference 41

5.5 cwdata.c File Reference 46

5.6 decrypt.c File Reference 47
5.7 dicho.c File Reference 51
5.8 dicho.h File Reference 56
5.9 encrypt.c File Reference 58
5.10 gf.c File Reference 60
5.11 gf.h File Reference 63
5.12 keypair.c File Reference 67
5.13 main_cwinfo.c File Reference 69
5.14 main_decrypt.c File Reference 70
5.15 main_encrypt.c File Reference 71
5.16 main_genparams.c File Reference 73
5.17 main_keygen.c File Reference 74
5.18 main_mce.c File Reference 75
5.19 main_secinfo.c File Reference 76
5.20 mat.c File Reference 77
5.21 matrix.h File Reference 79
5.22 mceliece.h File Reference 82
5.23 params.h File Reference 84
5.24 poly.c File Reference 85
5.25 poly.h File Reference 90
5.26 precomp.c File Reference 96
5.27 precomp.h File Reference 104
5.28 sizes.h File Reference 107
5.29 workfactor.c File Reference 110
5.30 workfactor.h File Reference 112

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

buff 13
code_arith 15
distrib_t 17
elt 18
leaf_info_t 20
lnode 21
matrix 22
polynome 24
precomp 25
tnode 27

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

arith.c . . . 29
arith.h . . . 32
buff.c . . . 35
buff.h . . . 41
cwdata.c . . . 46
decrypt.c . . . 47
dicho.c . . . 51
dicho.h . . . 56
encrypt.c . . . 58
gf.c . . . 60
gf.h . . . 63
keypair.c . . . 67
main_cwinfo.c . . . 69
main_decrypt.c . . . 70
main_encrypt.c . . . 71
main_genparams.c . . . 73
main_keygen.c . . . 74
main_mce.c . . . 75
main_secinfo.c . . . 76
mat.c . . . 77
matrix.h . . . 79
mceliece.h . . . 82
params.h . . . 84
poly.c . . . 85
poly.h . . . 90
precomp.c . . . 96
precomp.h . . . 104
sizes.h . . . 107
workfactor.c . . . 110
workfactor.h . . . 112

Chapter 3

The crypto-system

3.1 Introduction

It was introduced by Bob McEliece in 1978 [**?**] and is among the oldest public-key encryption scheme. It's security is related to hard algorithmic problems of algebraic coding theory whereas for most other public-key systems it is connected to algorithmic number theory (RSA, ECC, ...). Its main advantages are very efficient encryption and decryption procedures and a good practical and theoretical security. On the other hand, its main drawbacks are a public key of large size and a cyphertext which is larger than the cleartext.

3.1.1 General idea

The cleartext of k binary digits is encoded into a codeword of $n > k$ binary digits by mean of some public encoder of a linear code of length n and dimension k. The ciphertext is obtained by flipping t randomly chosen bits in this codeword.

If t is less than half the minimum Hamming distance of the linear code, to one ciphertext correspond only one possible cleartext. If n, k and t are large enough, computing the cleartext from the ciphertext is intractable, unless some side information on the algebraic structure of the code is known.

3.1.2 Description

Let $\mathcal{F}$ denote a family of binary linear codes of length n, dimension k for which a t-error correcting procedure is known.

Key generation: The legal user picks randomly and uniformly a code C in the family $\mathcal{F}$. Let G_0 be a generator matrix of C. The public key is equal to $G = SG_0P$ where S a random $k \times k$ non-singular binary matrix and P a random $n \times n$ permutation matrix.

Encryption: The cleartext is a word x of $\mathbf{F}_2^k$. The ciphertext is a word of $\mathbf{F}_2^n$ equal to $xG + e$ where e is randomly chosen with a Hamming weight t.

Decryption: The ciphertext is a word y of $\mathbf{F}_2^n$. The cleartext is recovered by applying the t-error correcting procedure of C to yP^{-1}.

In practice: Bob McEliece proposed the use of binary Goppa codes (see §??) with $m = 10$, $n = 1024$ and $t = 50$. The dimension is then $k = 524$. To keep up with 25 years of progress in algorithmics and computers, larger codes are required and we now need $m = 11$, $n = 2048$ and $30 \leq t \leq 120$. Using another family of linear codes is possible but must be done with great care since it has a significant impact on security. For instance using concatenated codes [?] or (generalized) Reed-Solomon codes [?] is unsafe.

3.2 Security

In this section, we assume that t-error correcting binary Goppa codes of length n and dimension k are being used. For a given set of parameters n, k and t, the two approaches for the cryptanalysis

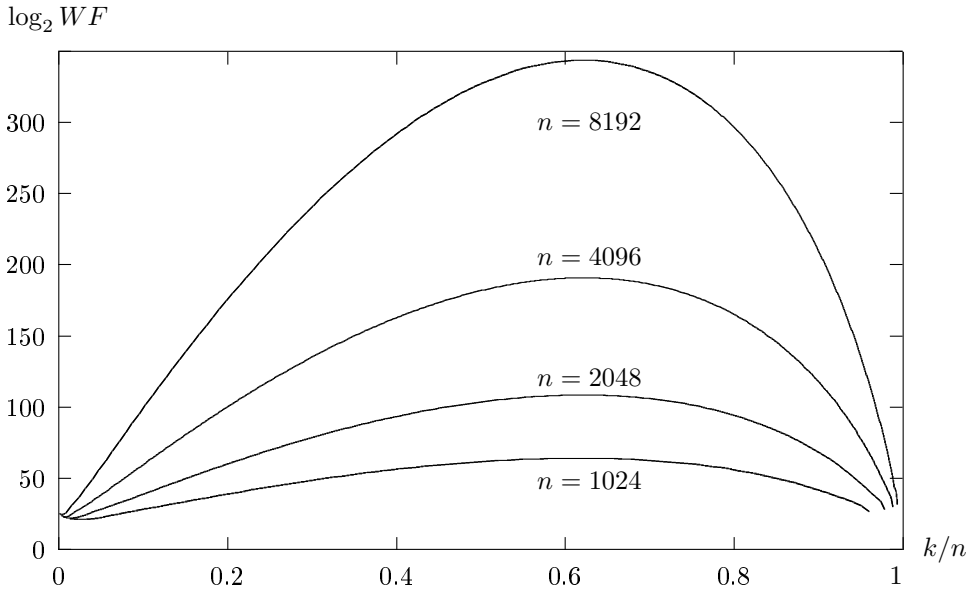


Figure 3.1: Binary work factor for the decoding attack of McEliece cryptosystem

are:

- the *decoding attack*: decode t error in a known binary linear code of length n and dimension k,
- the *structural attack*: deduce from the public key G an efficient t-error correcting procedure.

The decoding attack is related to the syndrome decoding problem [?] and rapidly becomes intractable when the parameters grow. In practice, for a fixed rate ($R = k/n$) the best known algorithms and implementations [?, ?] have a computation cost growing exponentially with t. The binary work factor, that is the average number of binary operations, required to correct t errors in a linear code of length n and transmission rate R is

$$WF(n, R, t) = P(n)2^{t \log_2 \frac{1}{1-R}}$$

where $P(n)$ roughly behaves as a polynomial in n of small degree (0 to 3 depending on the implementation). For a Goppa code $t = (n-k)/m = n(1-R)/m$ and the work factor can be written

$$WF(n, R, t) = P(n)2^{\frac{n}{m}(1-R)\log_2 \frac{1}{1-R}}.$$

The expected cost of the Canteaut-Chabaud algorithm [**?**] can be obtained by a Markov chain computation. An estimate for the binary work factor for decoding in various Goppa codes (whose structure is hidden) is given in Figure 3.1.

For the structural attack (with Goppa codes), nothing significantly better is known than enumerating all possible generator polynomial for a given support until one is found which is equal, up to a permutation, to the code generated by the public-key (this can be done by using the support splitting algorithm [**?**]). The cost grows exponentially with tm and is always higher than the cost of the decoding attack.

Choosing the parameters. We aim a work factor larger than 2^{85} binary operations for the best known attack. With Goppa codes of length $n = 2048$, we need a generator of degree $t \geq 30$. Table 3.1 presents some parameters and their main features.

n	1024	2048	**2048**	4096	2048
m	10	11	**11**	12	11
t	50	30	**32**	20	70
ciphertext size (in bits)	1024	2048	**2048**	4096	2048
message size (in bits)	524	1718	**1696**	3856	1278
information rate	0.51	0.84	**0.83**	0.94	0.62
public key size (in KB)	32	69	**73**	113	120
security exponent*	62.1	86.4	**89.2**	86.1	108.4

* *logarithm in base two of the binary work factor*

Table 3.1: Some parameters for the McEliece system

We will implement $(m, t) = (11, 32)$.

3.3 Binary Goppa codes

Let m be a positive integer, and let n and t be two positive integers such that $n \leq 2^m$ and $t < n/m$. A binary Goppa code $\Gamma(L, g)$ is defined by an ordered subset $L = (\alpha_1, \ldots, \alpha_n)$ of $\mathbf{F}_{2^m}$ of cardinality n, called *support*, and an irreducible[1] monic polynomial $g(z)$ of degree t in $\mathbf{F}_{2^m}[z]$, called *generator*. It consists of all words $a = (a_1, \ldots, a_n) \in \mathbf{F}_2^n$ such that $R_a(z) = 0$ where

$$R_a(z) = \sum_{j=1}^{n} \frac{a_j}{z - \alpha_j} \mod g(z). \tag{3.1}$$

This code is linear, has dimension[2] $k \geq n - tm$ and minimum distance $2t + 1$ at least. We denote $\mathcal{G}_{m,n,t}$ the set of all binary Goppa codes with a support of cardinality n in $\mathbf{F}_{2^m}$ and an irreducible generator of degree t over $\mathbf{F}_{2^m}$.

Proposition 1 *We have*

$$(a \in \Gamma(L, g)) \Leftrightarrow \left(a \cdot H_{L,g}^T = 0\right) \text{ where } H_{L,g} = \begin{pmatrix} \frac{1}{g(\alpha_1)} & \frac{1}{g(\alpha_2)} & \cdots & \frac{1}{g(\alpha_n)} \\ \frac{\alpha_1}{g(\alpha_1)} & \frac{\alpha_2}{g(\alpha_2)} & \cdots & \frac{\alpha_n}{g(\alpha_n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \frac{\alpha_2^{t-1}}{g(\alpha_2)} & \cdots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix}. \tag{3.2}$$

[1] square-free without roots in L in the most general definition

[2] for parameters suitable with the McEliece system, the equality always holds

The matrix $H_{L,g}$ is a parity check matrix of some particular generalized Reed-Solomon code. The binary Goppa code $\Gamma(L,g)$ is its binary subcode. A binary parity check matrix of $\Gamma(L,g)$ can be obtained by writing each element of $\mathbf{F}_{2^m}$ in a basis of $\mathbf{F}_{2^m}$ over $\mathbf{F}_2$. Each row of $H_{L,g}$ is replaced by m binary rows, corresponding to the coordinates in that basis. The corresponding matrix is binary, of size $tm \times n$, and is almost always full rank when t is not too large.

3.3.1 Algebraic decoding of Goppa codes

For any $e \in \{0,1\}$, we denote $\mathrm{supp}(e)$ the support of e (the non-zero positions) and we define the locator polynomial of e as $\sigma_e(z) = \prod_{j \in \mathrm{supp}(e)}(z - \alpha_j)$.

Proposition 2 *Let $b \in \{0,1\}^n$ such that $b = a + e$ with $a \in \Gamma(L,g)$ and $w_H(e) \leq t$. We consider the odd and even parts of the locator polynomial of e: $\sigma_e(z) = u(z)^2 + zv(z)^2$.*

1. *There is a unique polynomial $S(z)$ of degree $< t$ such that*

$$S(z)^2 = z + \frac{1}{R_b(z)} \mod g(z).$$

2. *The polynomials $u(z)$ and $v(z)$ are the unique solution (up to a multiplicative constant) of degree $\leq t/2$ and $\leq (t-1)/2$ respectively to the equation*

$$u(z) = S(z)v(z) \mod g(z).$$

Proof: From (3.1), and because $R_a(z) = 0$ we have $R_b(z) = R_e(z)$, and

$$R_b(z) = R_e(z) = \sum_{j=0}^{n} \frac{e_j}{z - \alpha_j} = \sum_{j \in \mathrm{supp}(e)} \frac{1}{z - \alpha_j} = \frac{\sigma_e'(z)}{\sigma_e(z)} \mod g(z) \tag{3.3}$$

where $\sigma_e'(z)$ is the derivative of $\sigma_e(z)$ (according to z). Because of the characteristic 2, we have $\sigma_e'(z) = v(z)^2$ and thus (3.3) can be rewritten as

$$R_b(z)u(z)^2 = (1 + zR_b(z))\, v(z)^2 \mod g(z). \tag{3.4}$$

Because $g(z)$ is irreducible, we can define the inverse $h(z)$ of $R_b(z)$ modulo $g(z)$, and we get

$$u(z)^2 = (h(z) + z)\, v(z)^2 \mod g(z). \tag{3.5}$$

Finally, the mapping following over the polynomials modulo $g(z)$

$$\begin{array}{rcl} \mathbf{F}_{2^m}[z]/(g(z)) & \to & \mathbf{F}_{2^m}[z]/(g(z)) \\ f(z) & \mapsto & f(z)^2 \end{array} \tag{3.6}$$

is bijective (and $\mathbf{F}_2$-linear). Thus there is a unique polynomial $S(z)$ such that $z + h(z) = S(z)^2 \mod g(z)$ and we have

$$u(z)^2 = S(z)^2 v(z)^2 \mod g(z).$$

Since $g(z)$ is square-free (it is even irreducible) we obtain a key equation verified by $u(z)$ and $v(z)$

$$u(z) = S(z)v(z) \mod g(z). \tag{3.7}$$

Because of the weight of the error is t or less, the polynomials $u(z)$ and $v(z)$ have degree at most $t/2$ and $(t-1)/2$ respectively. With those conditions on the degrees the solution to (3.7) is unique up to a multiplicative constant. $\Diamond$

In fact the proof of this proposition describes percisely the Patterson decoding algorithm [?]. The main steps of that algorithm are:

1. Compute the syndrome $R_b(z) = \sum_{j=1}^{n} b_j/(z - \alpha_j) \mod g(z)$.
2. Compute $h(z) = 1/R_b(z) \mod g(z)$ by the extended Euclidean algorithm.
3. Compute $S(z)$ such that $z + h(z) = S(z)^2 \mod g(z)$ which is done by precomputing the $T_i(z)^2 = z^i \mod g(z)$ for $i = 0, \ldots, t-1$. The $\mathbf{F}_2$-linearity is used on-line to compute $S(z)$ form the $T_i(z)$'s.
4. Solve the key equation (3.7) by the extended Euclidean algorithm.
5. Compute the roots of $\sigma_e(z) = u(z)^2 + zv(z)^2$ in $\mathbf{F}_{2^m}$.

3.4 Specification

3.4.1 A randomized version of McEliece

We consider an instance of McEliece of secret key $\mathcal{C} \in \mathcal{G}_{m,n,t}$ and public key G. Let us consider the two following mappings

$$\begin{array}{rccc|rccc} E_G: & \mathbf{F}_2^k \times W_{n,t} & \longrightarrow & \mathbf{F}_2^n & D_G: & \mathbf{F}_2^n & \longrightarrow & \mathbf{F}_2^k \times W_{n,t} \\ & (x,e) & \longmapsto & xG+e & & y & \longmapsto & \begin{cases} (x,e) & \text{if } y = xG+e \\ \text{FAIL} & \text{else} \end{cases} \end{array}$$

With those mappings, encryption and decryption can be defined as:

`encrypt`(x)	`decrypt`(y)
$e \leftarrow$ `random`$(W_{n,t})$	$(x,e) \leftarrow D_G(y)$
`return` $E_G(x,e)$	`return` x

Now, we will denote $PRNG(s,\ell)$ a binary word of length ℓ generated from the seed s by your favorite pseudo-random number generator. We define encryption and decryption as follows:

`encrypt`(x)	`decrypt`(y)
$e \leftarrow$ `random`$(W_{n,t})$	$(x,e) \leftarrow D_G(y)$
`return` $E_G(x + PRNG(e,k), e)$	`return` $x + PRNG(e,k)$

The above procedure are inverse of each other. Moreover, this allows the use of a systematic generator matrix which is unsafe otherwise (see [?, p. 34] for an example). Also, it prevents Berson's attack [?] and probably other similar "malleability flaws".

We will implement this version of the system. The PRNG will be the one provided by the EBATS package (Salsa).

3.4.2 Public key

Let $\Gamma(L,g) \in \mathcal{G}_{m,n,t}$ be an irreducible binary Goppa code. We denote $r = tm$ and we assume that it has dimension $k = n - tm$ exactly.

- Let H be the $r \times n$ binary matrix whose term in $(im+\ell)$-th row and j-th column is

$$b_\ell\left(\frac{\alpha_j^i}{g(\alpha_j)}\right), 0 \le i < t, 1 \le j \le n, 1 \le \ell \le m$$

where $b_\ell(\gamma) \in \mathbf{F}_2$ is the ℓ-th coordinate of $\gamma \in \mathbf{F}_{2^m}$ in some fixed basis of $\mathbf{F}_{2^m}$ over $\mathbf{F}_2$.

- We compute a systematic form $H_S = (R^T \mid I_r)$ of the $r \times n$ matrix H. If the last r columns of H are singular, we permute columns and change L in such a way that H_S has the prescribed form and is a parity check matrix of $\Gamma(L, g)$.

  I_r is the $r \times r$ identity matrix, R is a $k \times r$ binary matrix and R^T its transpose.

- Publish R as the public key ($G = (I_k \mid R)$ is a generator matrix of $\Gamma(L, g)$).

3.4.3 Encryption

Let $f_K()$ be a stream cipher parameterized by a key K. For any binary string x, we denote $f_K(x)$ its encryption. We assume it has the same length as x and that $f_K(f_K(x)) = x$.

Let $W_{n,t}$ denote the set of binary words of length n and Hamming weight t. Let $x \in \{0,1\}^k$ be the cleartext

- pick e at random in $W_{n,t}$,
- compute $x' = f_e(x)$,
- compute $y' = (x' \mid x'R^T)$,
- the ciphertext is $y = y' + e$.

3.4.4 Decryption

The decryption uses Patterson algorithm. Some values are precomputed

- For all α_j, $1 \leq j \leq n$
$$f_{\alpha_j}(z) = \frac{1}{z - \alpha_j} \mod g(z)$$
- for $i = 0, 1, \ldots, t-1$
$$T_i(z) = \sqrt{z^i} \mod g(z)$$

The ciphertext is $y = (y_1, \ldots, y_n) \in \{0,1\}^n$

- compute
$$R_e(z) = \sum_{j=1}^{n} \frac{y_j}{z - \alpha_j} \mod g(z) = \sum_{j=1}^{n} y_j f_{\alpha_j}(z),$$
- compute (using Euclidean algorithm)
$$h(z) = z + \frac{1}{R_e(z)} \mod g(z),$$
- compute (let $h(z) = h_0 + h_1 z + \ldots + h_{t-1} z^{t-1}$)
$$S(z) = \sqrt{h(z)} \mod g(z) = \sum_{i=0}^{t-1} h_i^{2^{m-1}} T_i(z),$$

- compute (using Euclidean algorithm) $u(z)$ and $v(z)$ of degree at most $t/2$ and $(t-1)/2$ respectively such that
$$u(z) = v(z)S(z) \mod g(z),$$
- compute the locator polynomial
$$\sigma(z) = u(z)^2 + zv(z)^2,$$
- compute the roots $(\alpha_{\ell_1}, \ldots, \alpha_{\ell_t})$ of $\sigma(z)$ and let $e \in \mathbf{F}_2^n$ with $\mathrm{supp}(e) = \{\ell_1, \ldots, \ell_t\}$.
- let $y' = y + e = (x' \mid x'')$ with x' of length k, and compute the cleartext $x = f_e(x')$.

Chapter 4

Data Structure Documentation

4.1 buff Struct Reference

`#include <buff.h>`

Data Fields

- int **size**
- unsigned long **val**
- unsigned char **masque_dernier**
- unsigned char * **message**
- int **fin**
- int **dernier**
- int **courant**
- int **lock**

4.1.1 Detailed Description

Definition at line 4 of file buff.h.

4.1.2 Field Documentation

4.1.2.1 int buff→size

Definition at line 5 of file buff.h.

Referenced by bfill(), bflush(), bflush_partiel(), blook(), bread(), bread_available(), bread_bit(), bread_changer_position(), bread_lock(), bread_position(), bread_retour(), breadinit(), bstep(), bwrite(), bwrite_available(), bwrite_bit(), bwrite_bits(), bwrite_changer_position(), bwrite_lock(), and bwriteinit().

4.1.2.2 unsigned long buff→val

Definition at line 6 of file buff.h.

Referenced by bfill(), bflush(), bflush_partiel(), blook(), bread(), bread_bit(), bread_changer_position(), bread_retour(), breadinit(), bwrite(), bwrite_bit(), bwrite_bits(), bwrite_changer_position(), and bwriteinit().

4.1.2.3 unsigned char buff→masque_dernier

Definition at line 7 of file buff.h.

Referenced by bread_decaler_fin(), bread_getchar(), breadinit(), bwrite_decaler_fin(), bwrite_putchar(), and bwriteinit().

4.1.2.4 unsigned char* buff→message

Definition at line 8 of file buff.h.

Referenced by bread_getchar(), breadinit(), bwrite_changer_position(), bwrite_putchar(), and bwriteinit().

4.1.2.5 int buff→fin

Definition at line 9 of file buff.h.

Referenced by bread_available(), bread_decaler_fin(), bread_unlocked(), breadinit(), bwrite_available(), bwrite_decaler_fin(), bwrite_unlocked(), bwriteinit(), dicho(), and dichoinv().

4.1.2.6 int buff→dernier

Definition at line 9 of file buff.h.

Referenced by bread_decaler_fin(), bread_getchar(), breadinit(), bwrite_decaler_fin(), bwrite_putchar(), and bwriteinit().

4.1.2.7 int buff→courant

Definition at line 9 of file buff.h.

Referenced by bflush_partiel(), bread_available(), bread_changer_position(), bread_getchar(), bread_lock(), bread_position(), bread_retour(), breadinit(), bwrite_available(), bwrite_changer_position(), bwrite_lock(), bwrite_putchar(), and bwriteinit().

4.1.2.8 int buff→lock

Definition at line 9 of file buff.h.

Referenced by bread_lock(), bread_unlocked(), breadinit(), bwrite_lock(), bwrite_unlocked(), and bwriteinit().

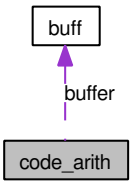
The documentation for this struct was generated from the following file:

- **buff.h**

4.2 code_arith Struct Reference

`#include <arith.h>`

Collaboration diagram for code_arith:



Data Fields

- int **compteur**
- unsigned long **min**
- unsigned long **max**
- struct **buff** * **buffer**

4.2.1 Detailed Description

Definition at line 16 of file arith.h.

4.2.2 Field Documentation

4.2.2.1 int code_arith→compteur

Definition at line 17 of file arith.h.

Referenced by ajuster(), arith_init(), coder(), coder_uniforme(), decoder(), decoder_uniforme(), and dicho().

4.2.2.2 unsigned long code_arith→min

Definition at line 18 of file arith.h.

Referenced by ajuster(), arith_init(), coder(), coder_uniforme(), decoder(), decoder_uniforme(), and dicho().

4.2.2.3 unsigned long code_arith→max

Definition at line 18 of file arith.h.

Referenced by ajuster(), arith_init(), coder(), coder_uniforme(), decoder(), and decoder_uniforme().

4.2.2.4 struct buff* code_arith→buffer [read]

Definition at line 19 of file arith.h.

Referenced by ajuster(), arith_init(), coder(), coder_uniforme(), decoder(), decoder_uniforme(), dicho(), dicho_b2cw(), dicho_cw2b(), and dichoinv().

The documentation for this struct was generated from the following file:

- **arith.h**

4.3 distrib_t Struct Reference

`#include <arith.h>`

Data Fields

- unsigned long **min**
- unsigned long **max**
- unsigned long * **prob**

4.3.1 Detailed Description

Definition at line 9 of file arith.h.

4.3.2 Field Documentation

4.3.2.1 unsigned long distrib_t

→

min

Definition at line 10 of file arith.h.

Referenced by clear_tree(), decoder(), dicho_build_tree(), dicho_si_lb_rec(), dicho_si_rec(), init_proba(), max_si_loss(), precomp_build(), tree_search(), and write_precomp().

4.3.2.2 unsigned long distrib_t

→

max

Definition at line 10 of file arith.h.

Referenced by clear_precomp(), clear_tree(), coder(), decoder(), dicho_build_tree(), dicho_si_lb_node(), dicho_si_lb_rec(), dicho_si_node(), dicho_si_rec(), init_proba(), max_si_loss(), precomp_build(), tree_search(), update_delta(), and write_precomp().

4.3.2.3 unsigned long* distrib_t

→

prob

Definition at line 11 of file arith.h.

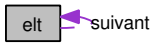
Referenced by decoder(), distrib_clear(), and init_proba().

The documentation for this struct was generated from the following file:

- **arith.h**

4.4 elt Struct Reference

Collaboration diagram for elt:



Data Fields

- int * **element**
- int **taille**
- int **nombre**
- int **pos**
- unsigned long **valeur**
- unsigned long **maximum**
- struct **elt** * **suivant**

4.4.1 Detailed Description

Definition at line 12 of file dicho.c.

4.4.2 Field Documentation

4.4.2.1 int* elt→element

Definition at line 13 of file dicho.c.

Referenced by dichoinv(), and dichoinv_rec().

4.4.2.2 int elt→taille

Definition at line 14 of file dicho.c.

Referenced by dicho(), dicho_rec(), dichoinv(), and dichoinv_rec().

4.4.2.3 int elt→nombre

Definition at line 14 of file dicho.c.

Referenced by dicho(), dicho_rec(), dichoinv(), and dichoinv_rec().

4.4.2.4 int elt→pos

Definition at line 15 of file dicho.c.

Referenced by dichoinv(), and dichoinv_rec().

4.4.2.5 unsigned long elt→valeur

Definition at line 16 of file dicho.c.

Referenced by dicho(), dicho_rec(), dichoinv(), and dichoinv_rec().

4.4.2.6 unsigned long elt→maximum

Definition at line 16 of file dicho.c.

Referenced by dicho(), dicho_rec(), dichoinv(), and dichoinv_rec().

4.4.2.7 struct elt* elt→suivant [read]

Definition at line 17 of file dicho.c.

Referenced by dicho(), dichoinv(), liste_alloc(), and liste_free().

The documentation for this struct was generated from the following file:

- **dicho.c**

4.5 leaf_info_t Struct Reference

`#include <precomp.h>`

Data Fields

- int **maximum**
- int **deadbits**

4.5.1 Detailed Description

Definition at line 7 of file precomp.h.

4.5.2 Field Documentation

4.5.2.1 int leaf_info_t→maximum

Definition at line 8 of file precomp.h.

Referenced by dicho_rec(), dichoinv_rec(), leaf_info(), and write_precomp().

4.5.2.2 int leaf_info_t→deadbits

Definition at line 8 of file precomp.h.

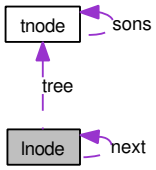
Referenced by dicho_rec(), dicho_si_from_list(), dicho_si_lb_rec(), dicho_si_rec(), dichoinv_rec(), leaf_info(), and write_precomp().

The documentation for this struct was generated from the following file:

- **precomp.h**

4.6 lnode Struct Reference

Collaboration diagram for lnode:



Data Fields

- **tree_t tree**
- struct **lnode** ∗ **next**

4.6.1 Detailed Description

Definition at line 15 of file precomp.c.

4.6.2 Field Documentation

4.6.2.1 tree_t lnode→tree

Definition at line 16 of file precomp.c.

Referenced by dicho_si_from_list(), list_alloc(), and tree_search().

4.6.2.2 struct lnode∗ lnode→next [read]

Definition at line 17 of file precomp.c.

Referenced by dicho_si_from_list(), list_alloc(), and tree_search().

The documentation for this struct was generated from the following file:

- **precomp.c**

4.7 matrix Struct Reference

`#include <matrix.h>`

Data Fields

- int **rown**
- int **coln**
- int **rwdcnt**
- int **alloc_size**
- unsigned long * **elem**

4.7.1 Detailed Description

Definition at line 6 of file matrix.h.

4.7.2 Field Documentation

4.7.2.1 int matrix→rown

Definition at line 7 of file matrix.h.

Referenced by key_genmat(), mat_copy(), mat_ini(), mat_ini_from_string(), mat_mul(), mat_rref(), and mat_vec_mul().

4.7.2.2 int matrix→coln

Definition at line 8 of file matrix.h.

Referenced by key_genmat(), mat_copy(), mat_ini(), mat_ini_from_string(), mat_mul(), and mat_rref().

4.7.2.3 int matrix→rwdcnt

Definition at line 9 of file matrix.h.

Referenced by mat_copy(), mat_ini(), mat_ini_from_string(), mat_rowxor(), and mat_vec_mul().

4.7.2.4 int matrix→alloc_size

Definition at line 10 of file matrix.h.

Referenced by keypair(), mat_ini(), mat_ini_from_string(), and mat_mul().

4.7.2.5 unsigned long* matrix→elem

Definition at line 11 of file matrix.h.

Referenced by keypair(), mat_copy(), mat_free(), mat_ini(), mat_ini_from_string(), mat_mul(), mat_rowxor(), and mat_vec_mul().

The documentation for this struct was generated from the following file:

- **matrix.h**

4.8 polynome Struct Reference

`#include <poly.h>`

Data Fields

- int **deg**
- int **size**
- **gf_t** * **coeff**

4.8.1 Detailed Description

Definition at line 6 of file poly.h.

4.8.2 Field Documentation

4.8.2.1 int polynome→deg

Definition at line 7 of file poly.h.

Referenced by poly_alloc(), poly_alloc_from_string(), poly_calcule_deg(), poly_copy(), poly_set(), poly_set_to_zero(), and poly_sqrtmod_init().

4.8.2.2 int polynome→size

Definition at line 7 of file poly.h.

Referenced by poly_alloc(), poly_alloc_from_string(), poly_calcule_deg(), poly_copy(), poly_set(), and poly_set_to_zero().

4.8.2.3 gf_t* polynome→coeff

Definition at line 8 of file poly.h.

Referenced by keypair(), poly_alloc(), poly_alloc_from_string(), poly_calcule_deg(), poly_copy(), poly_eval(), poly_free(), poly_set(), poly_set_to_zero(), poly_shiftmod(), and poly_sqrtmod_init().

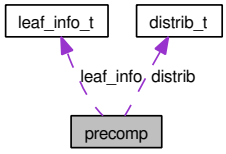
The documentation for this struct was generated from the following file:

- **poly.h**

4.9 precomp Struct Reference

`#include <precomp.h>`

Collaboration diagram for precomp:



Data Fields

- int **m**
- int **t**
- int **real_m**
- int **real_t**
- int * **offset**
- **distrib_t** ** **distrib**
- **leaf_info_t** ** **leaf_info**

4.9.1 Detailed Description

Definition at line 11 of file precomp.h.

4.9.2 Field Documentation

4.9.2.1 int precomp→m

Definition at line 12 of file precomp.h.

Referenced by clear_precomp(), dicho(), dicho_b2cw(), dicho_cw2b(), dicho_searchmin(), dicho_self_info_bounds(), dicho_si(), dicho_si_lb(), dichoinv(), precomp_build(), and write_precomp().

4.9.2.2 int precomp→t

Definition at line 12 of file precomp.h.

Referenced by clear_precomp(), dicho(), dicho_b2cw(), dicho_cw2b(), dicho_searchmin(), dicho_self_info_bounds(), dicho_si(), dicho_si_lb(), dichoinv(), precomp_build(), and write_precomp().

4.9.2.3 int precomp→real_m

Definition at line 12 of file precomp.h.

Referenced by dicho_b2cw(), dicho_cw2b(), dicho_searchmin(), dicho_self_info_bounds(), precomp_build(), and write_precomp().

4.9.2.4 int precomp→real_t

Definition at line 12 of file precomp.h.

Referenced by dicho_b2cw(), dicho_cw2b(), dicho_searchmin(), dicho_self_info_bounds(), precomp_build(), and write_precomp().

4.9.2.5 int* precomp→offset

Definition at line 13 of file precomp.h.

Referenced by clear_precomp(), and precomp_build().

4.9.2.6 distrib_t** precomp→distrib

Definition at line 14 of file precomp.h.

Referenced by clear_precomp(), precomp_build(), and write_precomp().

4.9.2.7 leaf_info_t** precomp→leaf_info

Definition at line 15 of file precomp.h.

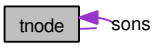
Referenced by dicho_rec(), dicho_si_from_list(), dicho_si_lb_rec(), dicho_si_rec(), dichoinv_rec(), precomp_build(), and write_precomp().

The documentation for this struct was generated from the following file:

- **precomp.h**

4.10 tnode Struct Reference

Collaboration diagram for tnode:



Data Fields

- int **m**
- int **t**
- struct **tnode** ** **sons**

4.10.1 Detailed Description

Definition at line 10 of file precomp.c.

4.10.2 Field Documentation

4.10.2.1 int tnode→m

Definition at line 11 of file precomp.c.

Referenced by clear_tree(), dicho_si_from_list(), leaf_alloc(), tree_alloc(), and tree_search().

4.10.2.2 int tnode→t

Definition at line 11 of file precomp.c.

Referenced by clear_tree(), dicho_si_from_list(), leaf_alloc(), tree_alloc(), and tree_search().

4.10.2.3 struct tnode** tnode→sons [read]

Definition at line 12 of file precomp.c.

Referenced by clear_tree(), leaf_alloc(), tree_alloc(), and tree_search().

The documentation for this struct was generated from the following file:

- **precomp.c**

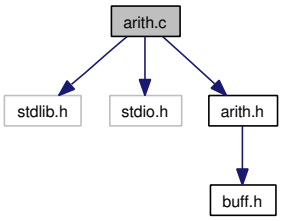
Chapter 5

File Documentation

5.1 arith.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "arith.h"
```

Include dependency graph for arith.c:



Functions

- int **l2** (unsigned long x)
- **arith_t arith_init** (struct **buff** *b)
- int **ajuster** (**arith_t** state, int coder)
- int **coder** (int i, **distrib_t** d, **arith_t** state)
- int **coder_uniforme** (unsigned long i, unsigned long n, **arith_t** state)
- int **chercher** (unsigned long valeur, unsigned long *sprob, int a, int b)
- int **decoder** (**distrib_t** d, int *lettre, **arith_t** state)
- unsigned long **decoder_uniforme** (unsigned long n, unsigned long *lettre, **arith_t** state)

5.1.1 Function Documentation

5.1.1.1 int ajuster (arith_t *state*, int *coder*)

Definition at line 62 of file arith.c.

References code_arith→buffer, bwrite(), bwrite_bit(), bwrite_bits(), code_arith→compteur, l2(), code_arith→max, code_arith→min, and PREC_INTER.

Referenced by coder(), coder_uniforme(), decoder(), and decoder_uniforme().

5.1.1.2 arith_t arith_init (struct buff * *b*)

Definition at line 49 of file arith.c.

References code_arith→buffer, code_arith→compteur, code_arith→max, code_arith→min, and PREC_INTER.

Referenced by dicho_b2cw(), and dicho_cw2b().

5.1.1.3 int chercher (unsigned long *valeur*, unsigned long * *sprob*, int *a*, int *b*)

Definition at line 189 of file arith.c.

Referenced by decoder().

5.1.1.4 int coder (int *i*, distrib_t *d*, arith_t *state*)

Definition at line 117 of file arith.c.

References ajuster(), code_arith→buffer, bwrite_lock(), code_arith→compteur, distrib_get_proba, distrib_t→max, code_arith→max, code_arith→min, PREC_INTER, and PREC_PROBA.

Referenced by dicho_rec().

5.1.1.5 int coder_uniforme (unsigned long *i*, unsigned long *n*, arith_t *state*)

Definition at line 156 of file arith.c.

References ajuster(), code_arith→buffer, bwrite_lock(), code_arith→compteur, code_arith→max, code_arith→min, and PREC_INTER.

Referenced by dicho().

5.1.1.6 int decoder (distrib_t *d*, int * *lettre*, arith_t *state*)

Definition at line 203 of file arith.c.

References ajuster(), blook(), bread_lock(), bstep(), code_arith→buffer, chercher(), code_arith→compteur, distrib_get_proba, distrib_t→max, code_arith→max, distrib_t→min, code_arith→min, PREC_INTER, PREC_PROBA, and distrib_t→prob.

Referenced by dichoinv_rec().

5.1.1.7 unsigned long decoder_uniforme (unsigned long *n*, unsigned long * *lettre*, arith_t *state*)

Definition at line 269 of file arith.c.

References ajuster(), blook(), bread_lock(), bstep(), code_arith→buffer, code_arith→compteur, code_arith→max, code_arith→min, and PREC_INTER.

Referenced by dichoinv().

5.1.1.8 int l2 (unsigned long *x*)

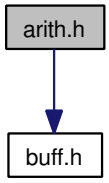
Definition at line 5 of file arith.c.

Referenced by adjust_delta(), ajuster(), and leaf_info().

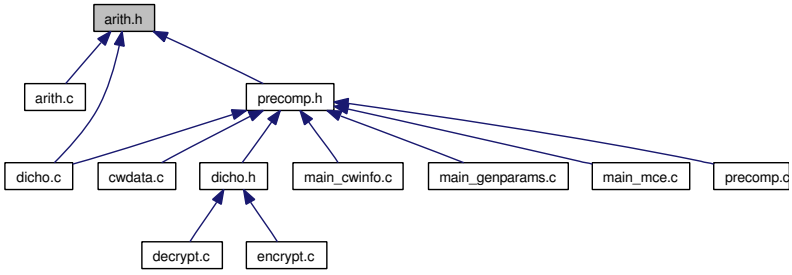
5.2 arith.h File Reference

```
#include "buff.h"
```

Include dependency graph for arith.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **distrib_t**
- struct **code_arith**

Defines

- #define **PREC_INTER** ((2 ∗ BUFFSIZE) / 3)
- #define **PREC_PROBA** (BUFFSIZE - PREC_INTER)
- #define **distrib_get_proba**(d, i) ((d).prob[(i) - (d).**min**])

Typedefs

- typedef struct **code_arith** ∗ **arith_t**

Functions

- **arith_t arith_init** (struct **buff** ∗b)
- int **coder** (int i, **distrib_t** d, **arith_t** state)
- int **coder_uniforme** (unsigned long i, unsigned long n, **arith_t** state)
- int **coder_bin_fin** (int i, **arith_t** state)
- int **decoder** (**distrib_t** d, int ∗lettre, **arith_t** state)
- unsigned long **decoder_uniforme** (unsigned long n, unsigned long ∗lettre, **arith_t** state)

- int **decoder_bin_fin** (**arith_t** state)
- int **tester_fin** (**arith_t** state)
- int **tester_compteur** (**arith_t** state)

5.2.1 Define Documentation

5.2.1.1 #define distrib_get_proba(d, i) ((d).prob[(i) - (d).min])

Definition at line 14 of file arith.h.

Referenced by coder(), decoder(), dicho_si_lb_node(), dicho_si_node(), init_proba(), update_delta(), and write_precomp().

5.2.1.2 #define PREC_INTER ((2 * BUFFSIZE) / 3)

Definition at line 6 of file arith.h.

Referenced by adjust_delta(), ajuster(), arith_init(), coder(), coder_uniforme(), decoder(), decoder_uniforme(), dicho_searchmin(), dicho_si_from_list(), dicho_si_lb_leaf(), dicho_si_lb_node(), and leaf_info().

5.2.1.3 #define PREC_PROBA (BUFFSIZE - PREC_INTER)

Definition at line 7 of file arith.h.

Referenced by coder(), decoder(), dicho(), dicho_si_lb_node(), dicho_si_node(), dichoinv(), init_proba(), leaf_info(), and update_delta().

5.2.2 Typedef Documentation

5.2.2.1 typedef struct code_arith * arith_t

5.2.3 Function Documentation

5.2.3.1 arith_t arith_init (struct buff * *b*)

Definition at line 49 of file arith.c.

References code_arith→buffer, code_arith→compteur, code_arith→max, code_arith→min, and PREC_INTER.

Referenced by dicho_b2cw(), and dicho_cw2b().

5.2.3.2 int coder (int *i*, distrib_t *d*, arith_t *state*)

Definition at line 117 of file arith.c.

References ajuster(), code_arith→buffer, bwrite_lock(), code_arith→compteur, distrib_get_proba, distrib_t→max, code_arith→max, code_arith→min, PREC_INTER, and PREC_PROBA.

Referenced by dicho_rec().

5.2.3.3 int coder_bin_fin (int *i*, arith_t *state*)

5.2.3.4 int coder_uniforme (unsigned long *i*, unsigned long *n*, arith_t *state*)

Definition at line 156 of file arith.c.

References ajuster(), code_arith→buffer, bwrite_lock(), code_arith→compteur, code_arith→max, code_arith→min, and PREC_INTER.

Referenced by dicho().

5.2.3.5 int decoder (distrib_t *d*, int * *lettre*, arith_t *state*)

Definition at line 203 of file arith.c.

References ajuster(), blook(), bread_lock(), bstep(), code_arith→buffer, chercher(), code_arith→compteur, distrib_get_proba, distrib_t→max, code_arith→max, distrib_t→min, code_arith→min, PREC_INTER, PREC_PROBA, and distrib_t→prob.

Referenced by dichoinv_rec().

5.2.3.6 int decoder_bin_fin (arith_t *state*)

5.2.3.7 unsigned long decoder_uniforme (unsigned long *n*, unsigned long * *lettre*, arith_t *state*)

Definition at line 269 of file arith.c.

References ajuster(), blook(), bread_lock(), bstep(), code_arith→buffer, code_arith→compteur, code_arith→max, code_arith→min, and PREC_INTER.

Referenced by dichoinv().

5.2.3.8 int tester_compteur (arith_t *state*)

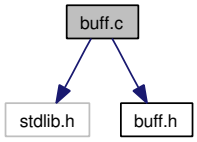
5.2.3.9 int tester_fin (arith_t *state*)

5.3 buff.c File Reference

```
#include <stdlib.h>
```

```
#include "buff.h"
```

Include dependency graph for buff.c:



Defines

- #define **LSB_TO_ONE**(i) ((i) ? ((1UL << (i)) - 1) : 0)
- #define **LSB_TO_ZERO**(i) (((i) == BUFFSIZE) ? 0 : (((unsigned long) -1) << (i)))

Functions

- unsigned char **bread_getchar** (**bread_t** bin)
- void **bwrite_putchar** (unsigned char c, **bwrite_t** bout)
- **bread_t breadinit** (unsigned char *message, int fin)
- **bwrite_t bwriteinit** (unsigned char *message, int fin)
- void **bfill** (**bread_t** bin)
- void **bflush** (**bwrite_t** bout)
- void **bflush_partiel** (**bwrite_t** bout)
- void **breadclose** (**bread_t** bin)
- void **bwriteclose** (**bwrite_t** bout)
- void **bread_retour** (**bread_t** bin)
- int **bread_available** (**bread_t** bin)
- int **bwrite_available** (**bwrite_t** bout)
- int **bread_unlocked** (**bread_t** bin)
- int **bwrite_unlocked** (**bwrite_t** bout)
- int **bread_position** (**bread_t** bin)
- void **bread_changer_position** (**bread_t** bin, int i)
- void **bread_decaler_fin** (**bread_t** bin, int i)
- void **bwrite_changer_position** (**bwrite_t** bout, int i)
- void **bwrite_decaler_fin** (**bwrite_t** bout, int i)
- unsigned **bread** (int i, **bread_t** bin)
- void **bread_lock** (int i, **bread_t** bin)
- void **bwrite_lock** (int i, **bwrite_t** bout)
- unsigned **blook** (int i, **bread_t** bin)
- void **bstep** (int i, **bread_t** bin)
- int **bread_bit** (**bread_t** bin)
- void **bwrite** (unsigned int x, int i, **bwrite_t** bout)
- void **bwrite_bit** (unsigned int x, **bwrite_t** bout)
- void **bwrite_bits** (unsigned int x, int n, **bwrite_t** bout)

5.3.1 Define Documentation

5.3.1.1 #define LSB_TO_ONE(i) ((i) ? ((1UL << (i)) - 1) : 0)

Definition at line 4 of file buff.c.

Referenced by bflush_partiel(), blook(), bread(), and bwrite().

5.3.1.2 #define LSB_TO_ZERO(i) (((i) == BUFFSIZE) ? 0 : (((unsigned long) -1) << (i)))

Definition at line 5 of file buff.c.

Referenced by bflush_partiel(), bread_decaler_fin(), breadinit(), bwrite_changer_position(), bwrite_decaler_fin(), and bwriteinit().

5.3.2 Function Documentation

5.3.2.1 void bfill (bread_t *bin*)

Definition at line 67 of file buff.c.

References bread_getchar(), BUFFSIZE, buff→size, and buff→val.

Referenced by bread(), bread_bit(), and bstep().

5.3.2.2 void bflush (bwrite_t *bout*)

Definition at line 78 of file buff.c.

References BUFFSIZE, bwrite_putchar(), buff→size, and buff→val.

Referenced by bwrite(), bwrite_bit(), and bwrite_bits().

5.3.2.3 void bflush_partiel (bwrite_t *bout*)

Definition at line 87 of file buff.c.

References bread_getchar(), BUFFSIZE, bwrite_putchar(), buff→courant, LSB_TO_ONE, LSB_TO_ZERO, buff→size, and buff→val.

Referenced by bwrite_changer_position(), and bwriteclose().

5.3.2.4 unsigned blook (int *i*, bread_t *bin*)

Definition at line 224 of file buff.c.

References bread_getchar(), LSB_TO_ONE, buff→size, and buff→val.

Referenced by decoder(), and decoder_uniforme().

5.3.2.5 unsigned bread (int *i*, bread_t *bin*)

Definition at line 199 of file buff.c.

References bfill(), LSB_TO_ONE, buff→size, and buff→val.

Referenced by dicho_b2cw(), and dichoinv().

5.3.2.6 int bread_available (bread_t *bin*)

Definition at line 130 of file buff.c.

References buff→courant, buff→fin, and buff→size.

5.3.2.7 int bread_bit (bread_t *bin*)

Definition at line 246 of file buff.c.

References bfill(), buff→size, and buff→val.

5.3.2.8 void bread_changer_position (bread_t *bin*, int *i*)

Definition at line 153 of file buff.c.

References bread_getchar(), buff→courant, buff→size, and buff→val.

Referenced by bread_decaler_fin(), dicho_b2cw(), and dichoinv().

5.3.2.9 void bread_decaler_fin (bread_t *bin*, int *i*)

Definition at line 163 of file buff.c.

References bread_changer_position(), bread_position(), buff→dernier, buff→fin, LSB_TO_ZERO, and buff→masque_dernier.

Referenced by dichoinv().

5.3.2.10 unsigned char bread_getchar (bread_t *bin*)

Definition at line 7 of file buff.c.

References buff→courant, buff→dernier, buff→masque_dernier, and buff→message.

Referenced by bfill(), bflush_partiel(), blook(), and bread_changer_position().

5.3.2.11 void bread_lock (int *i*, bread_t *bin*)

Definition at line 214 of file buff.c.

References buff→courant, buff→lock, and buff→size.

Referenced by decoder(), and decoder_uniforme().

5.3.2.12 int bread_position (bread_t *bin*)

Definition at line 149 of file buff.c.

References buff→courant, and buff→size.

Referenced by bread_decaler_fin().

5.3.2.13 void bread_retour (bread_t *bin*)

Definition at line 123 of file buff.c.

References buff→courant, buff→size, and buff→val.

5.3.2.14 int bread_unlocked (bread_t *bin*)

Definition at line 140 of file buff.c.

References buff→fin, and buff→lock.

Referenced by dichoinv().

5.3.2.15 void breadclose (bread_t *bin*)

Definition at line 114 of file buff.c.

Referenced by dicho_b2cw().

5.3.2.16 bread_t breadinit (unsigned char * *message*, int *fin*)

Definition at line 26 of file buff.c.

References buff→courant, buff→dernier, buff→fin, buff→lock, LSB_TO_ZERO, buff→masque_dernier, buff→message, buff→size, and buff→val.

Referenced by dicho_b2cw().

5.3.2.17 void bstep (int *i*, bread_t *bin*)

Definition at line 238 of file buff.c.

References bfill(), and buff→size.

Referenced by decoder(), and decoder_uniforme().

5.3.2.18 void bwrite (unsigned int *x*, int *i*, bwrite_t *bout*)

Definition at line 254 of file buff.c.

References bflush(), LSB_TO_ONE, buff→size, and buff→val.

Referenced by ajuster(), dicho(), and dicho_cw2b().

5.3.2.19 int bwrite_available (bwrite_t *bout*)

Definition at line 135 of file buff.c.

References BUFFSIZE, buff→courant, buff→fin, and buff→size.

5.3.2.20 void bwrite_bit (unsigned int *x*, bwrite_t *bout*)

Definition at line 267 of file buff.c.

References bflush(), buff→size, and buff→val.

Referenced by ajuster(), and dicho().

5.3.2.21 void bwrite_bits (unsigned int *x*, int *n*, bwrite_t *bout*)

Definition at line 275 of file buff.c.

References bflush(), BUFFSIZE, buff→size, and buff→val.

Referenced by ajuster(), and dicho().

5.3.2.22 void bwrite_changer_position (bwrite_t *bout*, int *i*)

Definition at line 170 of file buff.c.

References bflush_partiel(), BUFFSIZE, buff→courant, LSB_TO_ZERO, buff→message, buff→size, and buff→val.

Referenced by dicho(), and dicho_cw2b().

5.3.2.23 void bwrite_decaler_fin (bwrite_t *bout*, int *i*)

Definition at line 192 of file buff.c.

References buff→dernier, buff→fin, LSB_TO_ZERO, and buff→masque_dernier.

Referenced by dicho().

5.3.2.24 void bwrite_lock (int *i*, bwrite_t *bout*)

Definition at line 218 of file buff.c.

References BUFFSIZE, buff→courant, buff→lock, and buff→size.

Referenced by coder(), and coder_uniforme().

5.3.2.25 void bwrite_putchar (unsigned char *c*, bwrite_t *bout*)

Definition at line 16 of file buff.c.

References buff→courant, buff→dernier, buff→masque_dernier, and buff→message.

Referenced by bflush(), and bflush_partiel().

5.3.2.26 int bwrite_unlocked (bwrite_t *bout*)

Definition at line 145 of file buff.c.

References buff→fin, and buff→lock.

Referenced by dicho().

5.3.2.27 void bwriteclose (bwrite_t *bout*)

Definition at line 118 of file buff.c.

References bflush_partiel().

Referenced by dicho_cw2b().

5.3.2.28 bwrite_t bwriteinit (unsigned char * *message*, int *fin*)

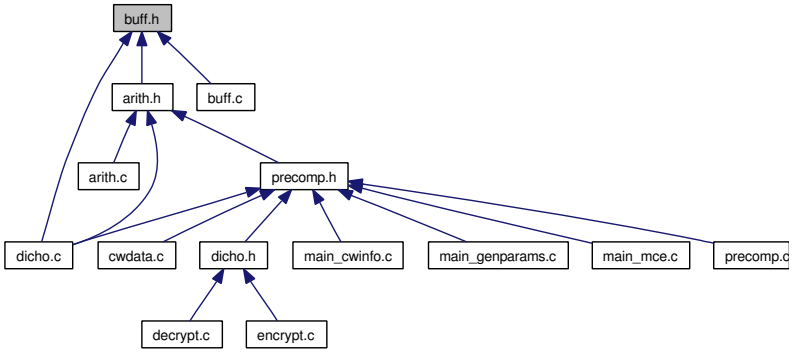
Definition at line 46 of file buff.c.

References BUFFSIZE, buff→courant, buff→dernier, buff→fin, buff→lock, LSB_TO_ZERO, buff→masque_dernier, buff→message, buff→size, and buff→val.

Referenced by dicho_cw2b().

5.4 buff.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct **buff**

Defines

- #define **BUFFSIZE** (8 ∗ sizeof (unsigned long))

Typedefs

- typedef struct **buff** ∗ **bread_t**
- typedef struct **buff** ∗ **bwrite_t**

Functions

- **bread_t breadinit** (unsigned char ∗message, int fin)
- **bwrite_t bwriteinit** (unsigned char ∗message, int fin)
- void **breadclose** (**bread_t** bin)
- void **bwriteclose** (**bwrite_t** bout)
- void **bread_retour** (**bread_t** bin)
- int **bread_available** (**bread_t** bin)
- int **bwrite_available** (**bwrite_t** bout)
- int **bread_unlocked** (**bread_t** bin)
- int **bwrite_unlocked** (**bwrite_t** bout)
- void **bread_changer_position** (**bread_t** bin, int i)
- void **bread_decaler_fin** (**bread_t** bin, int i)
- void **bwrite_changer_position** (**bwrite_t** bout, int i)
- void **bwrite_decaler_fin** (**bwrite_t** bout, int i)
- void **bread_lock** (int i, **bread_t** bin)
- void **bwrite_lock** (int i, **bwrite_t** bout)
- unsigned **bread** (int i, **bread_t** bin)
- unsigned **blook** (int i, **bread_t** bin)

- void **bstep** (int i, **bread_t** bin)
- int **bread_bit** (**bread_t** bin)
- void **bwrite** (unsigned int x, int i, **bwrite_t** bout)
- void **bwrite_bit** (unsigned int x, **bwrite_t** bout)
- void **bwrite_bits** (unsigned int x, int n, **bwrite_t** bout)

5.4.1 Define Documentation

5.4.1.1 #define BUFFSIZE (8 * sizeof (unsigned long))

Definition at line 12 of file buff.h.

Referenced by bfill(), bflush(), bflush_partiel(), bwrite_available(), bwrite_bits(), bwrite_changer_position(), bwrite_lock(), and bwriteinit().

5.4.2 Typedef Documentation

5.4.2.1 typedef struct buff* bread_t

Definition at line 14 of file buff.h.

5.4.2.2 typedef struct buff* bwrite_t

Definition at line 15 of file buff.h.

5.4.3 Function Documentation

5.4.3.1 unsigned blook (int *i*, bread_t *bin*)

Definition at line 224 of file buff.c.

References bread_getchar(), LSB_TO_ONE, buff→size, and buff→val.

Referenced by decoder(), and decoder_uniforme().

5.4.3.2 unsigned bread (int *i*, bread_t *bin*)

Definition at line 199 of file buff.c.

References bfill(), LSB_TO_ONE, buff→size, and buff→val.

Referenced by dicho_b2cw(), and dichoinv().

5.4.3.3 int bread_available (bread_t *bin*)

Definition at line 130 of file buff.c.

References buff→courant, buff→fin, and buff→size.

5.4.3.4 int bread_bit (bread_t *bin*)

Definition at line 246 of file buff.c.

References bfill(), buff→size, and buff→val.

5.4.3.5 void bread_changer_position (bread_t *bin*, int *i*)

Definition at line 153 of file buff.c.

References bread_getchar(), buff→courant, buff→size, and buff→val.

Referenced by bread_decaler_fin(), dicho_b2cw(), and dichoinv().

5.4.3.6 void bread_decaler_fin (bread_t *bin*, int *i*)

Definition at line 163 of file buff.c.

References bread_changer_position(), bread_position(), buff→dernier, buff→fin, LSB_TO_-ZERO, and buff→masque_dernier.

Referenced by dichoinv().

5.4.3.7 void bread_lock (int *i*, bread_t *bin*)

Definition at line 214 of file buff.c.

References buff→courant, buff→lock, and buff→size.

Referenced by decoder(), and decoder_uniforme().

5.4.3.8 void bread_retour (bread_t *bin*)

Definition at line 123 of file buff.c.

References buff→courant, buff→size, and buff→val.

5.4.3.9 int bread_unlocked (bread_t *bin*)

Definition at line 140 of file buff.c.

References buff→fin, and buff→lock.

Referenced by dichoinv().

5.4.3.10 void breadclose (bread_t *bin*)

Definition at line 114 of file buff.c.

Referenced by dicho_b2cw().

5.4.3.11 bread_t breadinit (unsigned char * *message*, int *fin*)

Definition at line 26 of file buff.c.

References buff→courant, buff→dernier, buff→fin, buff→lock, LSB_TO_ZERO, buff→masque_dernier, buff→message, buff→size, and buff→val.

Referenced by dicho_b2cw().

5.4.3.12 void bstep (int *i*, bread_t *bin*)

Definition at line 238 of file buff.c.

References bfill(), and buff→size.

Referenced by decoder(), and decoder_uniforme().

5.4.3.13 void bwrite (unsigned int *x*, int *i*, bwrite_t *bout*)

Definition at line 254 of file buff.c.

References bflush(), LSB_TO_ONE, buff→size, and buff→val.

Referenced by ajuster(), dicho(), and dicho_cw2b().

5.4.3.14 int bwrite_available (bwrite_t *bout*)

Definition at line 135 of file buff.c.

References BUFFSIZE, buff→courant, buff→fin, and buff→size.

5.4.3.15 void bwrite_bit (unsigned int *x*, bwrite_t *bout*)

Definition at line 267 of file buff.c.

References bflush(), buff→size, and buff→val.

Referenced by ajuster(), and dicho().

5.4.3.16 void bwrite_bits (unsigned int *x*, int *n*, bwrite_t *bout*)

Definition at line 275 of file buff.c.

References bflush(), BUFFSIZE, buff→size, and buff→val.

Referenced by ajuster(), and dicho().

5.4.3.17 void bwrite_changer_position (bwrite_t *bout*, int *i*)

Definition at line 170 of file buff.c.

References bflush_partiel(), BUFFSIZE, buff→courant, LSB_TO_ZERO, buff→message, buff→size, and buff→val.

Referenced by dicho(), and dicho_cw2b().

5.4.3.18 void bwrite_decaler_fin (bwrite_t *bout*, int *i*)

Definition at line 192 of file buff.c.

References buff→dernier, buff→fin, LSB_TO_ZERO, and buff→masque_dernier.

Referenced by dicho().

5.4.3.19 void bwrite_lock (int *i*, bwrite_t *bout*)

Definition at line 218 of file buff.c.

References BUFFSIZE, buff→courant, buff→lock, and buff→size.

Referenced by coder(), and coder_uniforme().

5.4.3.20 int bwrite_unlocked (bwrite_t *bout*)

Definition at line 145 of file buff.c.

References buff→fin, and buff→lock.

Referenced by dicho().

5.4.3.21 void bwriteclose (bwrite_t *bout*)

Definition at line 118 of file buff.c.

References bflush_partiel().

Referenced by dicho_cw2b().

5.4.3.22 bwrite_t bwriteinit (unsigned char * *message*, int *fin*)

Definition at line 46 of file buff.c.

References BUFFSIZE, buff→courant, buff→dernier, buff→fin, buff→lock, LSB_TO_ZERO, buff→masque_dernier, buff→message, buff→size, and buff→val.

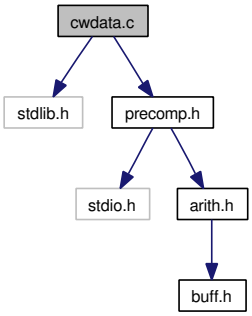
Referenced by dicho_cw2b().

5.5 cwdata.c File Reference

```
#include <stdlib.h>
```

```
#include "precomp.h"
```

Include dependency graph for cwdata.c:



Variables

- **precomp_t cwdata**

5.5.1 Variable Documentation

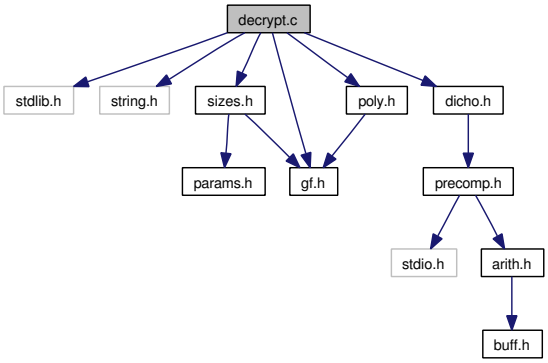
5.5.1.1 precomp_t cwdata

Definition at line 4 of file cwdata.c.

5.6 decrypt.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "sizes.h"
#include "gf.h"
#include "poly.h"
#include "dicho.h"
```

Include dependency graph for decrypt.c:



Functions

- void **sk_from_string** (const unsigned char *sk)
- void **sk_free** ()
- void **xor** (unsigned long *a, unsigned long *b)
- **poly_t syndrome** (const unsigned char *b)
- int **roots_berl_aux** (**poly_t** sigma, int d, **poly_t** *tr_aux, **poly_t** *tr, int e, **gf_t** *res)
- int **roots_berl** (**poly_t** sigma, **gf_t** *res)
- int **partition** (int *tableau, int gauche, int droite, int pivot)
- void **quickSort** (int *tableau, int gauche, int droite, int **min**, int max)
- int **decode** (const unsigned char *b, int *e)
- int **decrypt_block** (unsigned char *cleartext, unsigned char *ciphertext, const unsigned char *sk)

Variables

- **precomp_t cwdata**
- **poly_t g**
- **poly_t sqrtmod** [NB_ERRORS]
- **gf_t** * **Linv**
- unsigned long * **coeffs**

5.6.1 Function Documentation

5.6.1.1 int decode (const unsigned char * *b*, int * *e*)

Decodes one cyphertext `b` with Patterson's algorithm and places the error in `e`. Returns the number of errors if the decoding worked and -1 else.

References aux, EXT_DEGREE, gf_init(), gf_inv, gf_mul_fast, gf_sqrt, gf_square, gf_unit, gf_zero, Linv, NB_ERRORS, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, poly_deg, poly_eeaux(), poly_free(), poly_set_coeff, quickSort(), roots_berl(), and syndrome().

Referenced by decrypt_block().

5.6.1.2 int decrypt_block (unsigned char * *cleartext*, unsigned char * *ciphertext*, const unsigned char * *sk*)

Decrypts a block.

References BITS_TO_BYTES, decode(), dicho_cw2b(), DIMENSION, ERROR_SIZE, ERROR_WEIGHT, LOG_LENGTH, NB_ERRORS, sk_free(), and sk_from_string().

Referenced by main().

5.6.1.3 int partition (int * *tableau*, int *gauche*, int *droite*, int *pivot*)

Partition function for *Quicksort* algorithm.

Referenced by quickSort().

5.6.1.4 void quickSort (int * *tableau*, int *gauche*, int *droite*, int *min*, int *max*)

Quicksort algorithm.

References partition().

Referenced by decode().

5.6.1.5 int roots_berl (poly_t *sigma*, gf_t * *res*)

The Berle Kemp trace algorithm for root finding.

References EXT_DEGREE, gf_unit, NB_ERRORS, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, poly_free(), poly_set_coeff, poly_set_deg, poly_sqmod(), poly_sqmod_init(), and roots_berl_aux().

Referenced by decode().

5.6.1.6 int roots_berl_aux (poly_t *sigma*, int *d*, poly_t * *tr_aux*, poly_t * *tr*, int *e*, gf_t * *res*)

Auxiliary function for root finding.

References EXT_DEGREE, gf_div, gf_exp, gf_mul, gf_square, NB_ERRORS, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, poly_deg, poly_free(), poly_gcd(), and poly_quo().

Referenced by roots_berl().

5.6.1.7 void sk_free ()

Frees whatever has been allocated in `sk_from_string()`.

References NB_ERRORS.

Referenced by decrypt_block().

5.6.1.8 void sk_from_string (const unsigned char * *sk*)

Takes as input a secret key formatted as a string (for EBATS) and, as a side effect, sets the global variables `poly_t g`, gf_t * L, `poly_t sqrtmod[NB_ERRORS]` and `gf_t * coeffs` to be respectively, the generator polynomial and the support of the Goppa code, and the precomputed values defined in §3.4.4.

References BITS_TO_LONG, CODIMENSION, coeffs, LENGTH, Linv, NB_ERRORS, poly_alloc_from_string(), and poly_set_deg.

Referenced by decrypt_block().

5.6.1.9 poly_t syndrome (const unsigned char * *b*)

Computes the syndrome $R_b(z)$ of the binary word `b`.

References BIT_SIZE_OF_LONG, BITS_TO_LONG, CODIMENSION, coeffs, EXT_DEGREE, LENGTH, NB_ERRORS, poly_alloc(), poly_calcule_deg(), poly_set_coeff, and xor().

Referenced by decode().

5.6.1.10 void xor (unsigned long * *a*, unsigned long * *b*)

Returns the xor of 2 *long* type strings.

References BITS_TO_LONG, and CODIMENSION.

Referenced by syndrome().

5.6.2 Variable Documentation

5.6.2.1 unsigned long* coeffs

Definition at line 12 of file decrypt.c.

Referenced by sk_from_string(), and syndrome().

5.6.2.2 precomp_t cwdata

Definition at line 4 of file cwdata.c.

5.6.2.3 poly_t g

Definition at line 10 of file decrypt.c.

Referenced by keypair(), and poly_randgen_irred().

5.6.2.4 gf_t* Linv

Definition at line 11 of file decrypt.c.

Referenced by decode(), keypair(), and sk_from_string().

5.6.2.5 poly_t sqrtmod[NB_ERRORS]

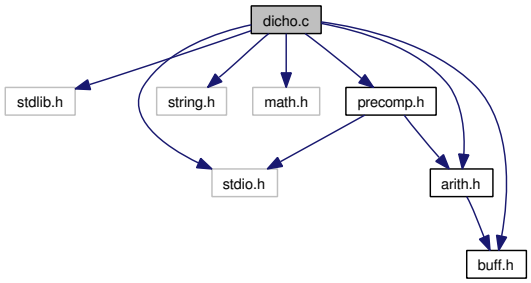
Definition at line 10 of file decrypt.c.

Referenced by keypair().

5.7 dicho.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "buff.h"
#include "arith.h"
#include "precomp.h"
```

Include dependency graph for dicho.c:



Data Structures

- struct **elt**

Typedefs

- typedef struct **elt** ∗ **liste_t**

Functions

- double **round** (double)
- int **l2** (unsigned long x)
- **liste_t liste_alloc** (**liste_t** s)
- void **liste_free** (**liste_t** l)
- int **is_leaf** (int m, int t)
- unsigned long **bino** (int a, int b)
- unsigned long **cw_coder** (int ∗res, int t)
- int **inv_bino** (unsigned long x, int t)
- void **cw_decoder** (unsigned long x, int t, int ∗res)
- int **dicho_rec** (int ∗cw, int i, int s, **arith_t** state, **precomp_t** p)
- int **dicho** (int ∗cw, **arith_t** state, **precomp_t** p)
- int **dichoinv_rec** (int ∗cw, int i, int s, int x, **arith_t** state, **precomp_t** p)
- int **dichoinv** (int ∗cw, **arith_t** state, **precomp_t** p)
- int **dicho_b2cw** (unsigned char ∗input_message, int ∗cw, int start, int len, int m, int t, **precomp_t** p)

- int **dicho_cw2b** (int *cw, unsigned char *output_message, int start, int len, int m, int t, **precomp_t** p)

Variables

- **liste_t liste_todo**
- **liste_t liste_inv**
- int * **aux**
- int **max_bino** [] = {0, 0, 0, 0, 0, 128, 64, 64, 32, 32, 32, 32, 32, 32, 32, 32, 32}
- unsigned long * **table_bino** []

5.7.1 Typedef Documentation

5.7.1.1 typedef struct elt * liste_t

5.7.2 Function Documentation

5.7.2.1 unsigned long bino (int *a*, int *b*)

Definition at line 144 of file dicho.c.

References table_bino.

5.7.2.2 unsigned long cw_coder (int * *res*, int *t*)

Definition at line 148 of file dicho.c.

References table_bino.

Referenced by dicho_rec().

5.7.2.3 void cw_decoder (unsigned long *x*, int *t*, int * *res*)

Definition at line 209 of file dicho.c.

References inv_bino(), round(), and table_bino.

Referenced by dichoinv().

5.7.2.4 int dicho (int * *cw*, arith_t *state*, precomp_t *p*)

Definition at line 327 of file dicho.c.

References aux, code_arith→buffer, bwrite(), bwrite_bit(), bwrite_bits(), bwrite_changer_position(), bwrite_decaler_fin(), bwrite_unlocked(), coder_uniforme(), code_arith→compteur, dicho_rec(), buff→fin, liste_free(), precomp→m, elt→maximum, code_arith→min, elt→nombre, PREC_PROBA, elt→suivant, precomp→t, elt→taille, and elt→valeur.

Referenced by dicho_cw2b().

5.7.2.5 int dicho_b2cw (unsigned char * *input_message*, int * *cw*, int *start*, int *len*, int *m*, int *t*, precomp_t *p*)

Definition at line 608 of file dicho.c.

References arith_init(), bread(), bread_changer_position(), breadclose(), breadinit(), code_arith→buffer, dichoinv(), precomp→m, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by encrypt_block().

5.7.2.6 int dicho_cw2b (int * *cw*, unsigned char * *output_message*, int *start*, int *len*, int *m*, int *t*, precomp_t *p*)

Definition at line 702 of file dicho.c.

References arith_init(), code_arith→buffer, bwrite(), bwrite_changer_position(), bwriteclose(), bwriteinit(), dicho(), precomp→m, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by decrypt_block().

5.7.2.7 int dicho_rec (int * *cw*, int *i*, int *s*, arith_t *state*, precomp_t *p*)

Definition at line 266 of file dicho.c.

References aux, coder(), cw_coder(), leaf_info_t→deadbits, is_leaf(), precomp→leaf_info, liste_alloc(), leaf_info_t→maximum, elt→maximum, elt→nombre, precomp_get_distrib, elt→taille, and elt→valeur.

Referenced by dicho().

5.7.2.8 int dichoinv (int * *cw*, arith_t *state*, precomp_t *p*)

Definition at line 469 of file dicho.c.

References bread(), bread_changer_position(), bread_decaler_fin(), bread_unlocked(), code_arith→buffer, cw_decoder(), decoder_uniforme(), dichoinv_rec(), elt→element, buff→fin, liste_free(), precomp→m, elt→maximum, elt→nombre, elt→pos, PREC_PROBA, elt→suivant, precomp→t, elt→taille, and elt→valeur.

Referenced by dicho_b2cw().

5.7.2.9 int dichoinv_rec (int * *cw*, int *i*, int *s*, int *x*, arith_t *state*, precomp_t *p*)

Definition at line 422 of file dicho.c.

References leaf_info_t→deadbits, decoder(), elt→element, is_leaf(), precomp→leaf_info, liste_alloc(), leaf_info_t→maximum, elt→maximum, elt→nombre, elt→pos, precomp_get_distrib, elt→taille, and elt→valeur.

Referenced by dichoinv().

5.7.2.10 int inv_bino (unsigned long *x*, int *t*)

Definition at line 187 of file dicho.c.

References max_bino, and table_bino.

Referenced by cw_decoder().

5.7.2.11 int is_leaf (int *m*, int *t*)

Definition at line 36 of file dicho.c.

Referenced by clear_precomp(), dicho_build_tree(), dicho_rec(), dicho_si_lb_rec(), dicho_si_rec(), dichoinv_rec(), precomp_build(), and write_precomp().

5.7.2.12 int l2 (unsigned long *x*)

Definition at line 5 of file arith.c.

Referenced by adjust_delta(), ajuster(), and leaf_info().

5.7.2.13 liste_t liste_alloc (liste_t *s*)

Definition at line 24 of file dicho.c.

References elt→suivant.

Referenced by dicho_rec(), and dichoinv_rec().

5.7.2.14 void liste_free (liste_t *l*)

Definition at line 30 of file dicho.c.

References elt→suivant.

Referenced by dicho(), and dichoinv().

5.7.2.15 double round (double)

Referenced by cw_decoder(), and init_proba().

5.7.3 Variable Documentation

5.7.3.1 int* aux

Definition at line 22 of file dicho.c.

Referenced by decode(), dicho(), dicho_rec(), memory_compl(), poly_eeaux(), and poly_sqrtmod_init().

5.7.3.2 liste_t liste_inv

Definition at line 21 of file dicho.c.

5.7.3.3 liste_t liste_todo

Definition at line 20 of file dicho.c.

5.7.3.4 int max_bino[] = {0, 0, 0, 0, 0, 128, 64, 64, 32, 32, 32, 32, 32, 32, 32, 32, 32}

Definition at line 48 of file dicho.c.

Referenced by inv_bino().

5.7.3.5 unsigned long* table_bino[]

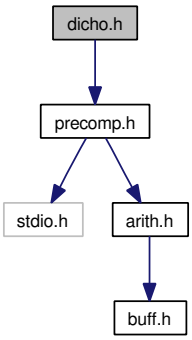
Definition at line 49 of file dicho.c.

Referenced by bino(), cw_coder(), cw_decoder(), and inv_bino().

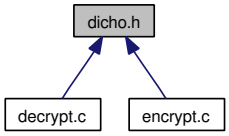
5.8 dicho.h File Reference

```
#include "precomp.h"
```

Include dependency graph for dicho.h:



This graph shows which files directly or indirectly include this file:



Functions

- int **dicho_b2cw** (unsigned char *input_message, int *cw, int start, int len, int m, int t, **precomp_t** p)
- int **dicho_cw2b** (int *cw, unsigned char *output_message, int start, int len, int m, int t, **precomp_t** p)

5.8.1 Function Documentation

5.8.1.1 int dicho_b2cw (unsigned char * *input_message*, int * *cw*, int *start*, int *len*, int *m*, int *t*, precomp_t *p*)

Definition at line 608 of file dicho.c.

References arith_init(), bread(), bread_changer_position(), breadclose(), breadinit(), code_arith→buffer, dichoinv(), precomp→m, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by encrypt_block().

5.8.1.2 int dicho_cw2b (int * *cw*, unsigned char * *output_message*, int *start*, int *len*, int *m*, int *t*, precomp_t *p*)

Definition at line 702 of file dicho.c.

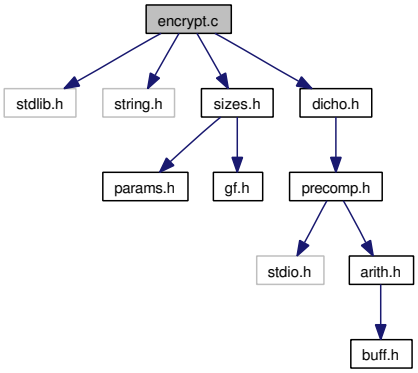
References arith_init(), code_arith→buffer, bwrite(), bwrite_changer_position(), bwriteclose(), bwriteinit(), dicho(), precomp→m, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by decrypt_block().

5.9 encrypt.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "sizes.h"
#include "dicho.h"
```

Include dependency graph for encrypt.c:



Functions

- void **vec_concat** (unsigned long *x, unsigned long *a, unsigned long *b)
- void **addto** (unsigned long *a, unsigned long *b)
- int **encrypt_block** (unsigned char *ciphertext, unsigned char *cleartext, const unsigned char *pk)

Variables

- **precomp_t cwdata**

5.9.1 Function Documentation

5.9.1.1 void addto (unsigned long * *a*, unsigned long * *b*)

Adds two *long* type arrays.

References BITS_TO_LONG, and CODIMENSION.

Referenced by encrypt_block().

5.9.1.2 int encrypt_block (unsigned char * *ciphertext*, unsigned char * *cleartext*, const unsigned char * *pk*)

Encrypts the message `m` using the public key `R`. The encrypted message is placed in `c`. Returns 1.

References addto(), BITS_TO_LONG, CODIMENSION, dicho_b2cw(), DIMENSION, ERROR_SIZE, ERROR_WEIGHT, LOG_LENGTH, NB_ERRORS, and vec_concat().

Referenced by main().

5.9.1.3 void vec_concat (unsigned long * *x*, unsigned long * *a*, unsigned long * *b*)

Copies the binary strings `a` and `b` into `x`. Assumes `la+lb` is a multiple of eight. When `la` and `b` are both multiples of eight, this is just a concatenation with the bytes of `a` coming first in `x`. If not, it does something complicated involving the middle byte (for future use maybe), which is not important as the whole program works only if `la` and `lb` are multiples of eight. A pointer to `x` is returned.

References BIT_SIZE_OF_LONG, BITS_TO_BYTES, CODIMENSION, and DIMENSION.

Referenced by encrypt_block().

5.9.2 Variable Documentation

5.9.2.1 precomp_t cwdata

Definition at line 4 of file cwdata.c.

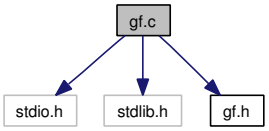
5.10 gf.c File Reference

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "gf.h"
```

Include dependency graph for gf.c:



Defines

- #define **MAX_EXT_DEG** 16

Functions

- void **gf_init_exp** ()
- void **gf_init_log** ()
- int **gf_init** (int extdeg)
- **gf_t gf_pow** (**gf_t** x, int i)
- **gf_t gf_rand** (int(*u8rnd)())

Variables

- static unsigned **prim_poly** [MAX_EXT_DEG+1]
- int **init_done** = 0

5.10.1 Define Documentation

5.10.1.1 #define MAX_EXT_DEG 16

The limit of extension degree.

Referenced by gf_init().

5.10.2 Function Documentation

5.10.2.1 int gf_init (int *extdeg*)

Redundant with gf.h???

References gf_cardinality, gf_exp, gf_extension_degree, gf_init_exp(), gf_init_log(), gf_log, gf_multiplicative_order, init_done, and MAX_EXT_DEG.

Referenced by decode(), and keypair().

5.10.2.2 void gf_init_exp ()

Redundant with gf.h???

References gf_exp, gf_extd, gf_ord, and prim_poly.

Referenced by gf_init().

5.10.2.3 void gf_init_log ()

Redundant with gf.h???

References gf_exp, gf_extd, gf_log, and gf_ord.

Referenced by gf_init().

5.10.2.4 gf_t gf_pow (gf_t *x*, int *i*)

Redundant with gf.h???

References gf_exp, gf_extd, gf_log, and gf_ord.

5.10.2.5 gf_t gf_rand (int(*)() *u8rnd*)

Redundant with gf.h???Definition at line 102 of file gf.c.

References gf_ord, and u8rnd().

Referenced by poly_randgen_irred().

5.10.3 Variable Documentation

5.10.3.1 int init_done = 0

Redundant with gf.h???

Referenced by gf_init().

5.10.3.2 unsigned prim_poly[MAX_EXT_DEG+1] [static]

Initial value:

```
{
   01,
   03,
   07,
   013,
   023,
   045,
   0103,
   0203,
   0435,
   01041,
   02011,
   04005,
   010123,
   020033,
```

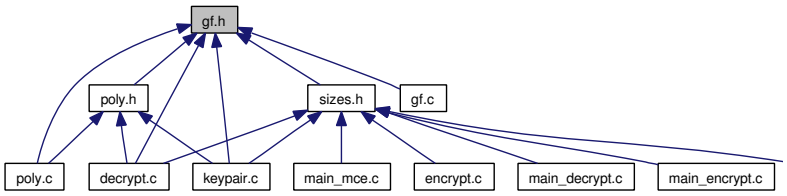
```
    042103,
    0100003,
    0210013

};
```

Hard coded values for irreducible polynomials.

Referenced by gf_init_exp().

5.11 gf.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define **gf_extd**() **gf_extension_degree**
- #define **gf_card**() **gf_cardinality**
- #define **gf_ord**() **gf_multiplicative_order**
- #define **gf_unit**() 1
- #define **gf_zero**() 0
- #define **gf_add**(x, y) ((x) ^ (y))
- #define **gf_exp**(i) **gf_exp**[i]
- #define **gf_log**(x) **gf_log**[x]
- #define **_gf_modq_1**(d) (((d) & gf_ord()) + ((d) >> gf_extd()))
- #define **gf_mul_fast**(x, y) ((y) ? **gf_exp**[_gf_modq_1(**gf_log**[x] + **gf_log**[y])] : 0)
- #define **gf_mul**(x, y) ((x) ? gf_mul_fast(x, y) : 0)
- #define **gf_square**(x) ((x) ? **gf_exp**[_gf_modq_1(**gf_log**[x] << 1)] : 0)
- #define **gf_sqrt**(x) ((x) ? **gf_exp**[_gf_modq_1(**gf_log**[x] << (gf_extd()-1))] : 0)
- #define **gf_div**(x, y) ((x) ? **gf_exp**[_gf_modq_1(**gf_log**[x] - **gf_log**[y])] : 0)
- #define **gf_inv**(x) **gf_exp**[gf_ord() - **gf_log**[x]]

Typedefs

- typedef unsigned short **gf_t**

Functions

- int **gf_init** (int extdeg)
- **gf_t gf_rand** (int(*u8rnd)())
- **gf_t gf_pow** (**gf_t** x, int i)

Variables

- int **gf_extension_degree**
- int **gf_cardinality**
- int **gf_multiplicative_order**
- **gf_t** * **gf_log**
- **gf_t** * **gf_exp**

5.11.1 Define Documentation

5.11.1.1 #define _gf_modq_1(d) (((d) & gf_ord()) + ((d) >> gf_extd()))

Returns a value between 0 and $(q-1)$ included, the class of 0 is represented by 0 or $q-1$ (this is why we write $\_K \rightarrow \exp[q-1] = \_K \rightarrow \exp[0] = 1$).

5.11.1.2 #define gf_add(x, y) ((x) ^ (y))

Adds 2 gf_t elements.

Referenced by poly_eval_aux(), poly_shiftmod(), and poly_syndrome_init().

5.11.1.3 #define gf_card() gf_cardinality

Not used.

5.11.1.4 #define gf_div(x, y) ((x) ? gf_exp[_gf_modq_1(gf_log[x] - gf_log[y])] : 0)

Divides x by y.

Referenced by poly_eeaux(), poly_shiftmod(), poly_syndrome_init(), and roots_berl_aux().

5.11.1.5 #define gf_exp(i) gf_exp[i]

Returns exponent.

Referenced by gf_init(), gf_init_exp(), gf_init_log(), gf_pow(), and roots_berl_aux().

5.11.1.6 #define gf_extd() gf_extension_degree

Returns $e$xtension degree of the field.

Referenced by gf_init_exp(), gf_init_log(), gf_pow(), poly_degppf(), and poly_sqrtmod_init().

5.11.1.7 #define gf_inv(x) gf_exp[gf_ord() - gf_log[x]]

Inverts the field element.

Referenced by decode(), key_genmat(), poly_quo(), and poly_rem().

5.11.1.8 #define gf_log(x) gf_log[x]

Returns logarithm.

Referenced by gf_init(), gf_init_log(), and gf_pow().

5.11.1.9 #define gf_mul(x, y) ((x) ? gf_mul_fast(x, y) : 0)

General multiplication.

Referenced by key_genmat(), poly_eval_aux(), poly_mul(), poly_shiftmod(), poly_syndrome_-init(), and roots_berl_aux().

5.11.1.10 #define gf_mul_fast(x, y) ((y) ? gf_exp[_gf_modq_1(gf_log[x] + gf_log[y])] : 0)

Multiplies 2 field elements excluding the 0 case for the first argument.

Referenced by decode(), poly_eeaux(), poly_quo(), poly_rem(), and poly_sqmod().

5.11.1.11 #define gf_ord() gf_multiplicative_order

Returns *o*rder of the multiplicative group of the field *i.e.* $q-1$.

Referenced by gf_init_exp(), gf_init_log(), gf_pow(), and gf_rand().

5.11.1.12 #define gf_sqrt(x) ((x) ? gf_exp[_gf_modq_1(gf_log[x] << (gf_extd()-1))] : 0)

Computes square-root.

Referenced by decode().

5.11.1.13 #define gf_square(x) ((x) ? gf_exp[_gf_modq_1(gf_log[x] << 1)] : 0)

Computes square.

Referenced by decode(), poly_sqmod(), and roots_berl_aux().

5.11.1.14 #define gf_unit() 1

Returns unit element in the field.

Referenced by decode(), poly_degppf(), poly_eeaux(), poly_randgen_irred(), poly_sqmod_-init(), poly_sqrtmod_init(), poly_syndrome_init(), and roots_berl().

5.11.1.15 #define gf_zero() 0

Returns zero element in the field.

Referenced by decode(), poly_calcule_deg(), poly_eeaux(), poly_eval_aux(), poly_quo(), poly_rem(), and poly_sqmod().

5.11.2 Typedef Documentation

5.11.2.1 typedef unsigned short gf_t

Data type to accommodate field structures up to extension degree 16.

5.11.3 Function Documentation

5.11.3.1 int gf_init (int *extdeg*)

Initializes the exponential and logarithmic values???.

References gf_cardinality, gf_exp, gf_extension_degree, gf_init_exp(), gf_init_log(), gf_log, gf_multiplicative_order, init_done, and MAX_EXT_DEG.

Referenced by decode(), and keypair().

5.11.3.2 gf_t gf_pow (gf_t *x*, int *i*)

Computes *i*nt power of the element *i.e.* x^i.

References gf_exp, gf_extd, gf_log, and gf_ord.

5.11.3.3 gf_t gf_rand (int(*)() *u8rnd*)

Random field element generator.

References gf_ord, and u8rnd().

Referenced by poly_randgen_irred().

5.11.4 Variable Documentation

5.11.4.1 int gf_cardinality

Cardinality.

Referenced by gf_init().

5.11.4.2 gf_t* gf_exp

Exponent.

5.11.4.3 int gf_extension_degree

Extension degree.

Referenced by gf_init().

5.11.4.4 gf_t* gf_log

Logarithm.

5.11.4.5 int gf_multiplicative_order

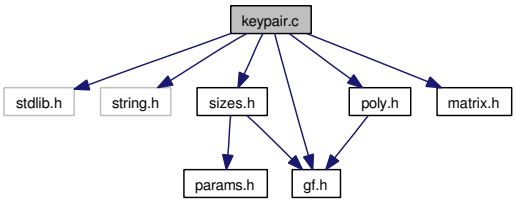
Multiplicative order *i.e.* $q-1$.

Referenced by gf_init().

5.12 keypair.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "sizes.h"
#include "gf.h"
#include "poly.h"
#include "matrix.h"
```

Include dependency graph for keypair.c:



Functions

- __inline int **u8rnd** ()
- __inline unsigned int **u32rnd** ()
- void **gop_supr** (int n, **gf_t** *L)
- **binmat_t key_genmat** (**gf_t** *L, **poly_t** g)
- int **keypair** (unsigned char *sk, unsigned char *pk)

5.12.1 Function Documentation

5.12.1.1 void gop_supr (int *n*, gf_t * *L*)

Generates a random support for the *Goppa code*.

References u32rnd().

Referenced by keypair().

5.12.1.2 binmat_t key_genmat (gf_t * *L*, poly_t *g*)

Generates a public key for a Goppa code given by its support L and its generator g. The Gaussian elimination of the generating matrix should be possible with the last columns. If not some columns are permuted and the support is changed accordingly.

References matrix

→

coln, EXT_DEGREE, gf_inv, gf_mul, LENGTH, mat_change_coeff, mat_coeff, mat_free(), mat_ini(), mat_rref(), mat_set_coeff_to_one, mat_set_to_zero, NB_ERRORS, poly_eval(), and matrix

→

rown.

Referenced by keypair().

5.12.1.3 int keypair (unsigned char * *sk*, unsigned char * *pk*)

Generates the key-pair.

References matrix

→

alloc_size, BIT_SIZE_OF_LONG, BITS_TO_LONG, CODIMENSION, polynome

→

coeff, matrix

→

elem, EXT_DEGREE, g, gf_init(), gop_supr(), key_genmat(), LENGTH, Linv, mat_free(), NB_ERRORS, poly_coeff, poly_free(), poly_randgen_irred(), poly_sqrtmod_init(), poly_-syndrome_init(), sqrtmod, and u8rnd().

Referenced by main().

5.12.1.4 __inline unsigned int u32rnd ()

32 bit randomizer.

References u8rnd().

Referenced by gop_supr().

5.12.1.5 __inline int u8rnd ()

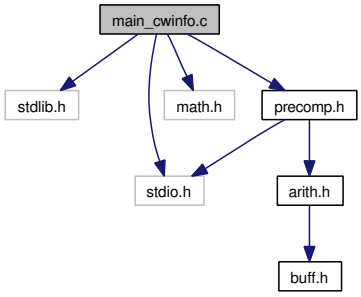
8 bit randomizer.

Referenced by gf_rand(), keypair(), poly_randgen_irred(), and u32rnd().

5.13 main_cwinfo.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "precomp.h"
```

Include dependency graph for main_cwinfo.c:



Functions

- int **main** (int argc, char **argv)

5.13.1 Function Documentation

5.13.1.1 int main (int *argc*, char ** *argv*)

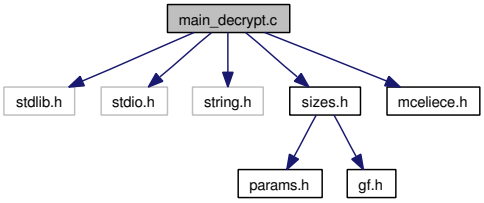
Definition at line 6 of file main_cwinfo.c.

References clear_precomp(), dicho_searchmin(), dicho_self_info_bounds(), log_binomial_d(), min, and precomp_build().

5.14 main_decrypt.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "sizes.h"
#include "mceliece.h"
```

Include dependency graph for main_decrypt.c:



Defines

- #define **DATA_BYTES** (CLEARTEXT_LENGTH / 8)

Functions

- int **main** (int argc, char **argv)

5.14.1 Define Documentation

5.14.1.1 #define DATA_BYTES (CLEARTEXT_LENGTH / 8)

Referenced by main().

5.14.2 Function Documentation

5.14.2.1 int main (int *argc*, char ** *argv*)

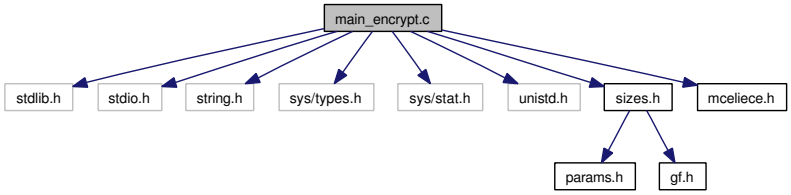
Definition at line 7 of file main_decrypt.c.

References CIPHERTEXT_BYTES, CLEARTEXT_BYTES, DATA_BYTES, decrypt_block(), EXT_DEGREE, NB_ERRORS, and SECRETKEY_BYTES.

5.15 main_encrypt.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include "sizes.h"
#include "mceliece.h"
```

Include dependency graph for main_encrypt.c:



Defines

- #define **DATA_BYTES** (CLEARTEXT_LENGTH / 8)

Functions

- __inline unsigned long long **rdtsc** ()
- unsigned char **padding** ()
- int **main** (int argc, char **argv)

5.15.1 Define Documentation

5.15.1.1 #define DATA_BYTES (CLEARTEXT_LENGTH / 8)

5.15.2 Function Documentation

5.15.2.1 int main (int *argc*, char ** *argv*)

Definition at line 22 of file main_encrypt.c.

References CIPHERTEXT_BYTES, CLEARTEXT_BYTES, DATA_BYTES, encrypt_block(), EXT_DEGREE, NB_ERRORS, padding(), and PUBLICKEY_BYTES.

5.15.2.2 unsigned char padding ()

Definition at line 18 of file main_encrypt.c.

Referenced by main().

5.15.2.3 __inline unsigned long long rdtsc ()

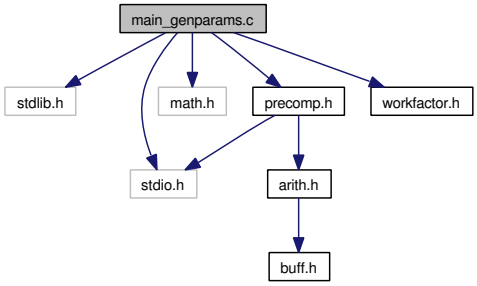
Definition at line 10 of file main_encrypt.c.

Referenced by main().

5.16 main_genparams.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "precomp.h"
#include "workfactor.h"
```

Include dependency graph for main_genparams.c:



Defines

- #define **LENGTH_LOSS** 1

Functions

- int **main** (int argc, char **argv)

5.16.1 Define Documentation

5.16.1.1 #define LENGTH_LOSS 1

Referenced by main().

5.16.2 Function Documentation

5.16.2.1 int main (int *argc*, char ** *argv*)

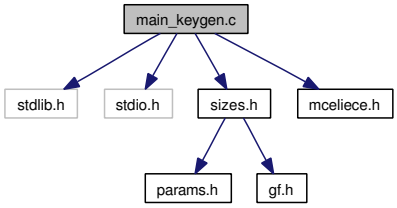
Definition at line 7 of file main_genparams.c.

References binomial_d(), clear_precomp(), dicho_self_info_bounds(), LENGTH_LOSS, precomp_build(), workfactor(), and write_precomp().

5.17 main_keygen.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "sizes.h"
#include "mceliece.h"
```

Include dependency graph for main_keygen.c:



Functions

- __inline unsigned long long **rdtsc** ()
- int **main** (int argc, char **argv)

5.17.1 Function Documentation

5.17.1.1 int main (int *argc*, char ** *argv*)

Definition at line 13 of file main_keygen.c.

References ERROR_WEIGHT, EXT_DEGREE, keypair(), LOG_LENGTH, NB_ERRORS, PUBLICKEY_BYTES, rdtsc(), and SECRETKEY_BYTES.

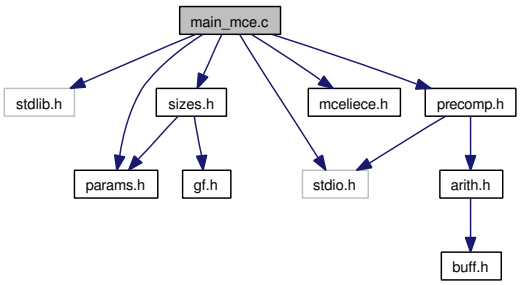
5.17.1.2 __inline unsigned long long rdtsc ()

Definition at line 6 of file main_keygen.c.

5.18 main_mce.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "sizes.h"
#include "mceliece.h"
#include "params.h"
#include "precomp.h"
```

Include dependency graph for main_mce.c:



Functions

- __inline unsigned long long **rdtsc** ()
- int **check** (unsigned char *cleartext, unsigned char *plaintext, int r)
- int **main** (int argc, char **argv)

5.18.1 Function Documentation

5.18.1.1 int check (unsigned char * *cleartext*, unsigned char * *plaintext*, int *r*)

Definition at line 15 of file main_mce.c.

References CLEARTEXT_BYTES, and CLEARTEXT_LENGTH.

Referenced by main().

5.18.1.2 int main (int *argc*, char ** *argv*)

Definition at line 43 of file main_mce.c.

References check(), CIPHERTEXT_BYTES, CLEARTEXT_BYTES, CLEARTEXT_LENGTH, decrypt_block(), encrypt_block(), ERROR_WEIGHT, keypair(), LOG_LENGTH, PUBLICKEY_BYTES, rdtsc(), and SECRETKEY_BYTES.

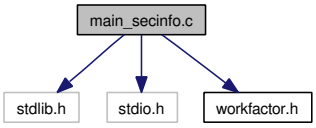
5.18.1.3 __inline unsigned long long rdtsc ()

Definition at line 8 of file main_mce.c.

5.19 main_secinfo.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "workfactor.h"
```

Include dependency graph for main_secinfo.c:



Functions

- int **main** (int argc, char **argv)

5.19.1 Function Documentation

5.19.1.1 int main (int *argc*, char ** *argv*)

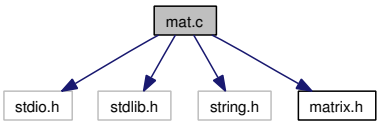
Definition at line 5 of file main_secinfo.c.

References workfactor().

5.20 mat.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "matrix.h"
```

Include dependency graph for mat.c:



Functions

- **binmat_t mat_ini** (int rown, int coln)
- **binmat_t mat_ini_from_string** (int rown, int coln, const unsigned char *s)
- void **mat_free** (**binmat_t** A)
- **binmat_t mat_copy** (**binmat_t** A)
- **binmat_t mat_rowxor** (**binmat_t** A, int a, int b)
- int * **mat_rref** (**binmat_t** A)
- void **mat_vec_mul** (unsigned long *cR, unsigned char *x, **binmat_t** A)
- **binmat_t mat_mul** (**binmat_t** A, **binmat_t** B)

5.20.1 Function Documentation

5.20.1.1 binmat_t mat_copy (binmat_t *A*)

Definition at line 46 of file mat.c.

References matrix→coln, matrix→elem, mat_ini(), matrix→rown, and matrix→rwdcnt.

5.20.1.2 void mat_free (binmat_t *A*)

Definition at line 40 of file mat.c.

References matrix→elem.

Referenced by key_genmat(), and keypair().

5.20.1.3 binmat_t mat_ini (int *rown*, int *coln*)

Definition at line 13 of file mat.c.

References matrix→alloc_size, BITS_PER_LONG, matrix→coln, matrix→elem, matrix→rown, and matrix→rwdcnt.

Referenced by key_genmat(), mat_copy(), and mat_mul().

5.20.1.4 binmat_t mat_ini_from_string (int *rown*, int *coln*, const unsigned char * *s*)

Definition at line 27 of file mat.c.

References matrix→alloc_size, BITS_PER_LONG, matrix→coln, matrix→elem, matrix→rown, and matrix→rwdcnt.

5.20.1.5 binmat_t mat_mul (binmat_t *A*, binmat_t *B*)

Definition at line 145 of file mat.c.

References matrix→alloc_size, matrix→coln, matrix→elem, mat_change_coeff, mat_coeff, mat_ini(), and matrix→rown.

5.20.1.6 binmat_t mat_rowxor (binmat_t *A*, int *a*, int *b*)

Definition at line 58 of file mat.c.

References matrix→elem, and matrix→rwdcnt.

Referenced by mat_rref().

5.20.1.7 int* mat_rref (binmat_t *A*)

Definition at line 71 of file mat.c.

References matrix→coln, mat_coeff, mat_rowxor(), and matrix→rown.

Referenced by key_genmat().

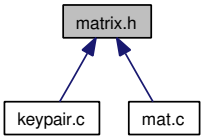
5.20.1.8 void mat_vec_mul (unsigned long * *cR*, unsigned char * *x*, binmat_t *A*)

Definition at line 128 of file mat.c.

References matrix→elem, matrix→rown, and matrix→rwdcnt.

5.21 matrix.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct **matrix**

Defines

- #define **BITS_PER_LONG** (8 * sizeof (unsigned long))
- #define **mat_coeff**(A, i, j) (((A) → elem[(i) * A → rwdcnt + (j) / BITS_PER_LONG] >> (j % BITS_PER_LONG)) & 1)
- #define **mat_set_coeff_to_one**(A, i, j) ((A) → elem[(i) * A → rwdcnt + (j) / BITS_PER_LONG] |= (1UL << ((j) % BITS_PER_LONG)))
- #define **mat_change_coeff**(A, i, j) ((A) → elem[(i) * A → rwdcnt + (j) / BITS_PER_LONG] ^= (1UL << ((j) % BITS_PER_LONG)))
- #define **mat_set_to_zero**(R) memset((R) → elem,0,(R) → alloc_size);

Typedefs

- typedef struct **matrix** * **binmat_t**

Functions

- **binmat_t mat_ini** (int rown, int coln)
- **binmat_t mat_ini_from_string** (int rown, int coln, const unsigned char *s)
- void **mat_free** (**binmat_t** A)
- **binmat_t mat_copy** (**binmat_t** A)
- **binmat_t mat_rowxor** (**binmat_t** A, int a, int b)
- int * **mat_rref** (**binmat_t** A)
- void **mat_vec_mul** (unsigned long *cR, unsigned char *x, **binmat_t** A)
- **binmat_t mat_mul** (**binmat_t** A, **binmat_t** B)

5.21.1 Define Documentation

5.21.1.1 #define BITS_PER_LONG (8 * sizeof (unsigned long))

Definition at line 4 of file matrix.h.

Referenced by mat_ini(), and mat_ini_from_string().

5.21.1.2 #define mat_change_coeff(A, i, j) ((A) → elem[(i) * A → rwdcnt + (j) / BITS_PER_LONG] ^= (1UL << ((j) % BITS_PER_LONG)))

Definition at line 17 of file matrix.h.

Referenced by key_genmat(), and mat_mul().

5.21.1.3 #define mat_coeff(A, i, j) (((A) → elem[(i) * A → rwdcnt + (j) / BITS_PER_LONG] >> (j % BITS_PER_LONG)) & 1)

Definition at line 14 of file matrix.h.

Referenced by key_genmat(), mat_mul(), and mat_rref().

5.21.1.4 #define mat_set_coeff_to_one(A, i, j) ((A) → elem[(i) * A → rwdcnt + (j) / BITS_PER_LONG] |= (1UL << ((j) % BITS_PER_LONG)))

Definition at line 16 of file matrix.h.

Referenced by key_genmat().

5.21.1.5 #define mat_set_to_zero(R) memset((R) → elem,0,(R) → alloc_size);

Definition at line 18 of file matrix.h.

Referenced by key_genmat().

5.21.2 Typedef Documentation

5.21.2.1 typedef struct matrix* binmat_t

5.21.3 Function Documentation

5.21.3.1 binmat_t mat_copy (binmat_t *A*)

Definition at line 46 of file mat.c.

References matrix→coln, matrix→elem, mat_ini(), matrix→rown, and matrix→rwdcnt.

5.21.3.2 void mat_free (binmat_t *A*)

Definition at line 40 of file mat.c.

References matrix→elem.

Referenced by key_genmat(), and keypair().

5.21.3.3 binmat_t mat_ini (int *rown*, int *coln*)

Definition at line 13 of file mat.c.

References matrix→alloc_size, BITS_PER_LONG, matrix→coln, matrix→elem, matrix→rown, and matrix→rwdcnt.

Referenced by key_genmat(), mat_copy(), and mat_mul().

5.21.3.4 binmat_t mat_ini_from_string (int *rown*, int *coln*, const unsigned char * *s*)

Definition at line 27 of file mat.c.

References matrix→alloc_size, BITS_PER_LONG, matrix→coln, matrix→elem, matrix→rown, and matrix→rwdcnt.

5.21.3.5 binmat_t mat_mul (binmat_t *A*, binmat_t *B*)

Definition at line 145 of file mat.c.

References matrix→alloc_size, matrix→coln, matrix→elem, mat_change_coeff, mat_coeff, mat_ini(), and matrix→rown.

5.21.3.6 binmat_t mat_rowxor (binmat_t *A*, int *a*, int *b*)

Definition at line 58 of file mat.c.

References matrix→elem, and matrix→rwdcnt.

Referenced by mat_rref().

5.21.3.7 int* mat_rref (binmat_t *A*)

Definition at line 71 of file mat.c.

References matrix→coln, mat_coeff, mat_rowxor(), and matrix→rown.

Referenced by key_genmat().

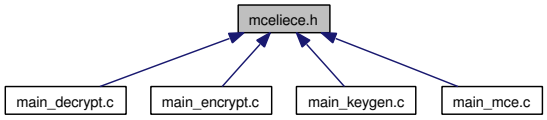
5.21.3.8 void mat_vec_mul (unsigned long * *cR*, unsigned char * *x*, binmat_t *A*)

Definition at line 128 of file mat.c.

References matrix→elem, matrix→rown, and matrix→rwdcnt.

5.22 mceliece.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- int **encrypt_block** (unsigned char *ciphertext, unsigned char *cleartext, const unsigned char *pk)
- int **decrypt_block** (unsigned char *cleartext, unsigned char *ciphertext, const unsigned char *sk)
- int **keypair** (unsigned char *sk, unsigned char *pk)

5.22.1 Function Documentation

5.22.1.1 int decrypt_block (unsigned char * *cleartext*, unsigned char * *ciphertext*, const unsigned char * *sk*)

Definition at line 271 of file decrypt.c.

References BITS_TO_BYTES, decode(), dicho_cw2b(), DIMENSION, ERROR_SIZE, ERROR_WEIGHT, LOG_LENGTH, NB_ERRORS, sk_free(), and sk_from_string().

Referenced by main().

5.22.1.2 int encrypt_block (unsigned char * *ciphertext*, unsigned char * *cleartext*, const unsigned char * *pk*)

Definition at line 41 of file encrypt.c.

References addto(), BITS_TO_LONG, CODIMENSION, dicho_b2cw(), DIMENSION, ERROR_SIZE, ERROR_WEIGHT, LOG_LENGTH, NB_ERRORS, and vec_concat().

Referenced by main().

5.22.1.3 int keypair (unsigned char * *sk*, unsigned char * *pk*)

Definition at line 92 of file keypair.c.

References matrix

→

alloc_size, BIT_SIZE_OF_LONG, BITS_TO_LONG, CODIMENSION, polynome

→

coeff, matrix

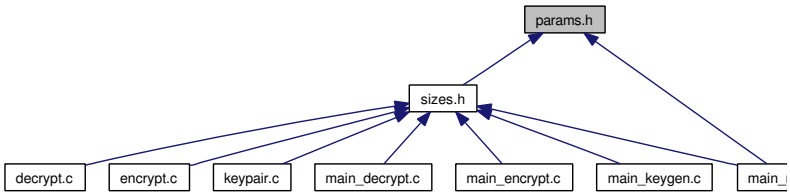
→

elem, EXT_DEGREE, g, gf_init(), gop_supr(), key_genmat(), LENGTH, Linv, mat_free(), NB_ERRORS, poly_coeff, poly_free(), poly_randgen_irred(), poly_sqrtmod_init(), poly_-syndrome_init(), sqrtmod, and u8rnd().

Referenced by main().

5.23 params.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define **LOG_LENGTH** 11
- #define **ERROR_WEIGHT** 79
- #define **REDUC** 0
- #define **ERROR_SIZE** 477

5.23.1 Define Documentation

5.23.1.1 #define ERROR_SIZE 477

Definition at line 5 of file params.h.

Referenced by decrypt_block(), and encrypt_block().

5.23.1.2 #define ERROR_WEIGHT 79

Definition at line 2 of file params.h.

Referenced by decrypt_block(), encrypt_block(), and main().

5.23.1.3 #define LOG_LENGTH 11

Definition at line 1 of file params.h.

Referenced by decrypt_block(), encrypt_block(), and main().

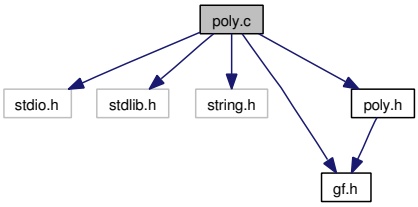
5.23.1.4 #define REDUC 0

Definition at line 4 of file params.h.

5.24 poly.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gf.h"
#include "poly.h"
```

Include dependency graph for poly.c:



Functions

- **poly_t poly_alloc** (int d)
- **poly_t poly_alloc_from_string** (int d, const unsigned char *s)
- **poly_t poly_copy** (**poly_t** p)
- void **poly_free** (**poly_t** p)
- void **poly_set_to_zero** (**poly_t** p)
- int **poly_calcule_deg** (**poly_t** p)
- void **poly_set** (**poly_t** p, **poly_t** q)
- **gf_t poly_eval_aux** (**gf_t** *coeff, **gf_t** a, int d)
- **poly_t poly_mul** (**poly_t** p, **poly_t** q)
- **gf_t poly_eval** (**poly_t** p, **gf_t** a)
- void **poly_rem** (**poly_t** p, **poly_t g**)
- void **poly_sqmod_init** (**poly_t g**, **poly_t** *sq)
- void **poly_sqmod** (**poly_t** res, **poly_t** p, **poly_t** *sq, int d)
- **poly_t poly_gcd_aux** (**poly_t** p1, **poly_t** p2)
- **poly_t poly_gcd** (**poly_t** p1, **poly_t** p2)
- **poly_t poly_quo** (**poly_t** p, **poly_t** d)
- int **poly_degppf** (**poly_t g**)
- void **poly_eeaux** (**poly_t** *u, **poly_t** *v, **poly_t** p, **poly_t g**, int t)
- **poly_t poly_randgen_irred** (int t, int(*u8rnd)())
- void **poly_shiftmod** (**poly_t** p, **poly_t g**)
- **poly_t** * **poly_sqrtmod_init** (**poly_t g**)
- **poly_t** * **poly_syndrome_init** (**poly_t** generator, **gf_t** *support, int n)

5.24.1 Function Documentation

5.24.1.1 poly_t poly_alloc (int *d*)

Definition at line 12 of file poly.c.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by decode(), poly_degppf(), poly_eeaux(), poly_mul(), poly_quo(), poly_randgen_irred(), poly_sqrtmod_init(), poly_syndrome_init(), roots_berl(), roots_berl_aux(), and syndrome().

5.24.1.2 poly_t poly_alloc_from_string (int *d*, const unsigned char * *s*)

Definition at line 23 of file poly.c.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by sk_from_string().

5.24.1.3 int poly_calcule_deg (poly_t *p*)

Definition at line 54 of file poly.c.

References polynome→coeff, polynome→deg, gf_zero, and polynome→size.

Referenced by decode(), poly_degppf(), poly_mul(), poly_quo(), poly_set(), poly_sqrtmod_init(), roots_berl(), roots_berl_aux(), and syndrome().

5.24.1.4 poly_t poly_copy (poly_t *p*)

Definition at line 33 of file poly.c.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by poly_gcd(), and poly_quo().

5.24.1.5 int poly_degppf (poly_t *g*)

Definition at line 231 of file poly.c.

References gf_extd, gf_unit, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_deg, poly_free(), poly_gcd(), poly_set_coeff, poly_set_deg, poly_sqmod(), and poly_sqmod_init().

Referenced by poly_randgen_irred().

5.24.1.6 void poly_eeaux (poly_t * *u*, poly_t * *v*, poly_t *p*, poly_t *g*, int *t*)

Definition at line 279 of file poly.c.

References aux, gf_div, gf_mul_fast, gf_unit, gf_zero, poly_addto_coeff, poly_alloc(), poly_coeff, poly_deg, poly_free(), poly_set(), poly_set_coeff, poly_set_deg, and poly_set_to_zero().

Referenced by decode().

5.24.1.7 gf_t poly_eval (poly_t *p*, gf_t *a*)

Definition at line 105 of file poly.c.

References polynome→coeff, poly_deg, and poly_eval_aux().

Referenced by key_genmat().

5.24.1.8 gf_t poly_eval_aux (gf_t * *coeff*, gf_t *a*, int *d*)

Definition at line 76 of file poly.c.

References gf_add, gf_mul, and gf_zero.

Referenced by poly_eval().

5.24.1.9 void poly_free (poly_t *p*)

Definition at line 44 of file poly.c.

References polynome→coeff.

Referenced by decode(), keypair(), poly_degppf(), poly_eeaux(), poly_gcd(), poly_quo(), poly_sqrtmod_init(), roots_berl(), and roots_berl_aux().

5.24.1.10 poly_t poly_gcd (poly_t *p1*, poly_t *p2*)

Definition at line 191 of file poly.c.

References poly_copy(), poly_deg, poly_free(), and poly_gcd_aux().

Referenced by poly_degppf(), and roots_berl_aux().

5.24.1.11 poly_t poly_gcd_aux (poly_t *p1*, poly_t *p2*)

Definition at line 182 of file poly.c.

References poly_deg, and poly_rem().

Referenced by poly_gcd().

5.24.1.12 poly_t poly_mul (poly_t *p*, poly_t *q*)

Definition at line 88 of file poly.c.

References gf_mul, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, and poly_deg.

5.24.1.13 poly_t poly_quo (poly_t *p*, poly_t *d*)

Definition at line 205 of file poly.c.

References gf_inv, gf_mul_fast, gf_zero, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, poly_copy(), poly_free(), poly_set_coeff, and poly_set_deg.

Referenced by roots_berl_aux().

5.24.1.14 poly_t poly_randgen_irred (int *t*, int(*)() *u8rnd*)

Definition at line 345 of file poly.c.

References g, gf_rand(), gf_unit, poly_alloc(), poly_degppf(), poly_set_coeff, poly_set_deg, and u8rnd().

Referenced by keypair().

5.24.1.15 void poly_rem (poly_t *p*, poly_t *g*)

Definition at line 110 of file poly.c.

References gf_inv, gf_mul_fast, gf_zero, poly_addto_coeff, poly_coeff, poly_deg, poly_set_coeff, poly_set_deg, and poly_tete.

Referenced by poly_gcd_aux(), and poly_sqmod_init().

5.24.1.16 void poly_set (poly_t *p*, poly_t *q*)

Definition at line 63 of file poly.c.

References polynome→coeff, polynome→deg, poly_calcule_deg(), and polynome→size.

Referenced by poly_eeaux(), and poly_sqrtmod_init().

5.24.1.17 void poly_set_to_zero (poly_t *p*)

Definition at line 49 of file poly.c.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by poly_eeaux(), poly_sqmod(), poly_sqmod_init(), and poly_sqrtmod_init().

5.24.1.18 void poly_shiftmod (poly_t *p*, poly_t *g*)

Definition at line 365 of file poly.c.

References polynome→coeff, gf_add, gf_div, gf_mul, and poly_deg.

Referenced by poly_sqrtmod_init().

5.24.1.19 void poly_sqmod (poly_t *res*, poly_t *p*, poly_t * *sq*, int *d*)

Definition at line 156 of file poly.c.

References gf_mul_fast, gf_square, gf_zero, poly_addto_coeff, poly_coeff, poly_deg, poly_set_coeff, poly_set_deg, and poly_set_to_zero().

Referenced by poly_degppf(), poly_sqrtmod_init(), and roots_berl().

5.24.1.20 void poly_sqmod_init (poly_t *g*, poly_t * *sq*)

Definition at line 131 of file poly.c.

References gf_unit, poly_deg, poly_rem(), poly_set_coeff, poly_set_deg, and poly_set_to_zero().

Referenced by poly_degppf(), poly_sqrtmod_init(), and roots_berl().

5.24.1.21 poly_t* poly_sqrtmod_init (poly_t *g*)

Definition at line 376 of file poly.c.

References aux, polynome→coeff, polynome→deg, gf_extd, gf_unit, poly_alloc(), poly_calcule_deg(), poly_deg, poly_free(), poly_set(), poly_set_coeff, poly_set_deg, poly_set_to_zero(), poly_shiftmod(), poly_sqmod(), and poly_sqmod_init().

Referenced by keypair().

5.24.1.22 poly_t* poly_syndrome_init (poly_t *generator*, gf_t * *support*, int *n*)

Definition at line 428 of file poly.c.

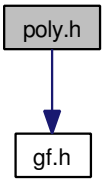
References gf_add, gf_div, gf_mul, gf_unit, poly_alloc(), poly_coeff, poly_deg, and poly_set_coeff.

Referenced by keypair().

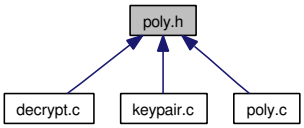
5.25 poly.h File Reference

```
#include "gf.h"
```

Include dependency graph for poly.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **polynome**

Defines

- #define **TRUE** 1
- #define **FALSE** 0
- #define **poly_deg**(p) ((p) → deg)
- #define **poly_size**(p) ((p) → size)
- #define **poly_set_deg**(p, d) ((p) → deg = (d))
- #define **poly_coeff**(p, i) ((p) → coeff[i])
- #define **poly_set_coeff**(p, i, a) ((p) → coeff[i] = (a))
- #define **poly_addto_coeff**(p, i, a) ((p) → coeff[i] = gf_add((p) → coeff[i], (a)))
- #define **poly_multo_coeff**(p, i, a) ((p) → coeff[i] = gf_mul((p) → coeff[i], (a)))
- #define **poly_tete**(p) ((p) → coeff[(p) → deg])

Typedefs

- typedef struct **polynome** ∗ **poly_t**

Functions

- int **poly_calcule_deg** (**poly_t** p)
- **poly_t poly_alloc** (int d)
- **poly_t poly_alloc_from_string** (int d, const unsigned char ∗s)
- **poly_t poly_copy** (**poly_t** p)
- void **poly_free** (**poly_t** p)

- void **poly_set_to_zero** (**poly_t** p)
- void **poly_set** (**poly_t** p, **poly_t** q)
- **poly_t poly_mul** (**poly_t** p, **poly_t** q)
- void **poly_rem** (**poly_t** p, **poly_t g**)
- void **poly_sqmod_init** (**poly_t g**, **poly_t** *sq)
- void **poly_sqmod** (**poly_t** res, **poly_t** p, **poly_t** *sq, int d)
- **poly_t poly_gcd** (**poly_t** p1, **poly_t** p2)
- **poly_t poly_quo** (**poly_t** p, **poly_t** d)
- **gf_t poly_eval** (**poly_t** p, **gf_t** a)
- int **poly_degppf** (**poly_t g**)
- void **poly_eeaux** (**poly_t** *u, **poly_t** *v, **poly_t** p, **poly_t g**, int t)
- **poly_t** * **poly_syndrome_init** (**poly_t** generator, **gf_t** *support, int n)
- **poly_t** * **poly_sqrtmod_init** (**poly_t g**)
- **poly_t poly_randgen_irred** (int t, int(*u8rnd)())

5.25.1 Define Documentation

5.25.1.1 #define FALSE 0

The false case.

5.25.1.2 #define poly_addto_coeff(p, i, a) ((p) → coeff[i] = gf_add((p) → coeff[i], (a)))

Adds `a` to the coefficient of degree `i` in `p`.

Referenced by decode(), poly_degppf(), poly_eeaux(), poly_mul(), poly_quo(), poly_rem(), poly_sqmod(), roots_berl(), and roots_berl_aux().

5.25.1.3 #define poly_coeff(p, i) ((p) → coeff[i])

Returns the coefficient of degree `i` in `p`.

Referenced by decode(), keypair(), poly_eeaux(), poly_mul(), poly_quo(), poly_rem(), poly_sqmod(), poly_syndrome_init(), roots_berl(), and roots_berl_aux().

5.25.1.4 #define poly_deg(p) ((p) → deg)

Returns the value of the degree field of `p`.

Referenced by decode(), poly_degppf(), poly_eeaux(), poly_eval(), poly_gcd(), poly_gcd_aux(), poly_mul(), poly_rem(), poly_shiftmod(), poly_sqmod(), poly_sqmod_init(), poly_sqrtmod_init(), poly_syndrome_init(), and roots_berl_aux().

5.25.1.5 #define poly_multo_coeff(p, i, a) ((p) → coeff[i] = gf_mul((p) → coeff[i], (a)))

Multiplies `a` to the coefficient of degree `i` in `p`.

5.25.1.6 #define poly_set_coeff(p, i, a) ((p) → coeff[i] = (a))

Sets the coefficient of degree `i` in `p` to `a`.

Referenced by decode(), poly_degppf(), poly_eeaux(), poly_quo(), poly_randgen_irred(), poly_rem(), poly_sqmod(), poly_sqmod_init(), poly_sqrtmod_init(), poly_syndrome_init(), roots_berl(), and syndrome().

5.25.1.7 #define poly_set_deg(p, d) ((p) → deg = (d))

Sets the value of the degree field of `p` to `d`.

Referenced by poly_degppf(), poly_eeaux(), poly_quo(), poly_randgen_irred(), poly_rem(), poly_sqmod(), poly_sqmod_init(), poly_sqrtmod_init(), roots_berl(), and sk_from_string().

5.25.1.8 #define poly_size(p) ((p) → size)

Returns the size of the polynomial `p`.

5.25.1.9 #define poly_tete(p) ((p) → coeff[(p) → deg])

Returns the coefficient of the highest degree of `p`.

Referenced by poly_rem().

5.25.1.10 #define TRUE 1

Computes the remainder into a new allocated polynomial. Degrees are used but not checked (to be corrected!).

5.25.2 Typedef Documentation

5.25.2.1 typedef struct polynome * poly_t

5.25.3 Function Documentation

5.25.3.1 poly_t poly_alloc (int *d*)

Allocates and returns a polynomial of size `d+1`. The coefficients are set to zero and the degree to -1.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by decode(), poly_degppf(), poly_eeaux(), poly_mul(), poly_quo(), poly_randgen_irred(), poly_sqrtmod_init(), poly_syndrome_init(), roots_berl(), roots_berl_aux(), and syndrome().

5.25.3.2 poly_t poly_alloc_from_string (int *d*, const unsigned char * *s*)

Allocates and returns a polynomial of size `d+1`. No allocation is made for the coefficients, `s` is used instead. The degree is set to -1 and is probably wrong. Added for efficiency. Do not use `poly_free()`.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by sk_from_string().

5.25.3.3 int poly_calcule_deg (poly_t *p*)

Computes the actual degree of `p`, sets the degree field to it and returns it.

References polynome→coeff, polynome→deg, gf_zero, and polynome→size.

Referenced by decode(), poly_degppf(), poly_mul(), poly_quo(), poly_set(), poly_sqrtmod_init(), roots_berl(), roots_berl_aux(), and syndrome().

5.25.3.4 poly_t poly_copy (poly_t *p*)

Copies a polynomial into another.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by poly_gcd(), and poly_quo().

5.25.3.5 int poly_degppf (poly_t *g*)

Returns the degree of the smallest factor.

References gf_extd, gf_unit, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_deg, poly_free(), poly_gcd(), poly_set_coeff, poly_set_deg, poly_sqmod(), and poly_sqmod_init().

Referenced by poly_randgen_irred().

5.25.3.6 void poly_eeaux (poly_t * *u*, poly_t * *v*, poly_t *p*, poly_t *g*, int *t*)

Returns a new allocated random monic irreducible polynomial of degree `t`. The second argument `u8rand()` should generate random bytes.

References aux, gf_div, gf_mul_fast, gf_unit, gf_zero, poly_addto_coeff, poly_alloc(), poly_coeff, poly_deg, poly_free(), poly_set(), poly_set_coeff, poly_set_deg, and poly_set_to_zero().

Referenced by decode().

5.25.3.7 gf_t poly_eval (poly_t *p*, gf_t *a*)

Evaluates `p` for a value `a` of the indeterminate. Degree is used but not checked (to be corrected!).

References polynome→coeff, poly_deg, and poly_eval_aux().

Referenced by key_genmat().

5.25.3.8 void poly_free (poly_t *p*)

Frees a polynomial proviously allocated by `poly_alloc()`.

References polynome→coeff.

Referenced by decode(), keypair(), poly_degppf(), poly_eeaux(), poly_gcd(), poly_quo(), poly_sqrtmod_init(), roots_berl(), and roots_berl_aux().

5.25.3.9 poly_t poly_gcd (poly_t *p1*, poly_t *p2*)

Returns the *gcd* of 2 polynomials `p1` and `p2`.

References poly_copy(), poly_deg, poly_free(), and poly_gcd_aux().

Referenced by poly_degppf(), and roots_berl_aux().

5.25.3.10 poly_t poly_mul (poly_t *p*, poly_t *q*)

Multiplies 2 polynomials `p` and `q`.

References gf_mul, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, and poly_deg.

5.25.3.11 poly_t poly_quo (poly_t *p*, poly_t *d*)

Returns the quotient of the division of `p` by `q`.

References gf_inv, gf_mul_fast, gf_zero, poly_addto_coeff, poly_alloc(), poly_calcule_deg(), poly_coeff, poly_copy(), poly_free(), poly_set_coeff, and poly_set_deg.

Referenced by roots_berl_aux().

5.25.3.12 poly_t poly_randgen_irred (int *t*, int(*)() *u8rnd*)

Returns a new allocated random monic irreducible polynomial of degree `t`. The second argument `u8rand()` should generate random bytes.

References g, gf_rand(), gf_unit, poly_alloc(), poly_degppf(), poly_set_coeff, poly_set_deg, and u8rnd().

Referenced by keypair().

5.25.3.13 void poly_rem (poly_t *p*, poly_t *g*)

Returns the remainder of the division of `p` by `q`.

References gf_inv, gf_mul_fast, gf_zero, poly_addto_coeff, poly_coeff, poly_deg, poly_set_coeff, poly_set_deg, and poly_tete.

Referenced by poly_gcd_aux(), and poly_sqmod_init().

5.25.3.14 void poly_set (poly_t *p*, poly_t *q*)

Copy `q` in `p`, redefinition of function *poly_ copy*.

References polynome→coeff, polynome→deg, poly_calcule_deg(), and polynome→size.

Referenced by poly_eeaux(), and poly_sqrtmod_init().

5.25.3.15 void poly_set_to_zero (poly_t *p*)

Sets all the coefficients of `p` to 0.

References polynome→coeff, polynome→deg, and polynome→size.

Referenced by poly_eeaux(), poly_sqmod(), poly_sqmod_init(), and poly_sqrtmod_init().

5.25.3.16 void poly_sqmod (poly_t *res*, poly_t *p*, poly_t * *sq*, int *d*)

Modulo p square of a certain polynomial g, sq[] contains the square modulo g of the base canonical polynomials of degree $< d$, where d is the degree of G. The table sq[] will be calculated by poly_sqmod_init().

References gf_mul_fast, gf_square, gf_zero, poly_addto_coeff, poly_coeff, poly_deg, poly_set_coeff, poly_set_deg, and poly_set_to_zero().

Referenced by poly_degppf(), poly_sqrtmod_init(), and roots_berl().

5.25.3.17 void poly_sqmod_init (poly_t *g*, poly_t * *sq*)

Returns a table of new allocated polynomials. The table size is the degree of g. The i-th entry is the square of z^i modulo $g(z)$.

References gf_unit, poly_deg, poly_rem(), poly_set_coeff, poly_set_deg, and poly_set_to_zero().

Referenced by poly_degppf(), poly_sqrtmod_init(), and roots_berl().

5.25.3.18 poly_t* poly_sqrtmod_init (poly_t *g*)

Returns a table of new allocated polynomials. The table size is the degree of g. The i-th entry is the square root of z^i modulo $g(z)$.

References aux, polynome→coeff, polynome→deg, gf_extd, gf_unit, poly_alloc(), poly_calcule_deg(), poly_deg, poly_free(), poly_set(), poly_set_coeff, poly_set_deg, poly_set_to_zero(), poly_shiftmod(), poly_sqmod(), and poly_sqmod_init().

Referenced by keypair().

5.25.3.19 poly_t* poly_syndrome_init (poly_t *generator*, gf_t * *support*, int *n*)

Returns a table of new allocated polynomials. The table size is equal to n which should also be the size of support. The i-th entry is the inverse of the polynomial $z -$ support[i] modulo the polynomial generator.

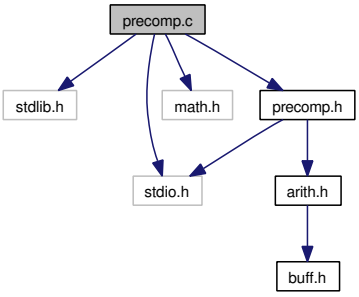
References gf_add, gf_div, gf_mul, gf_unit, poly_alloc(), poly_coeff, poly_deg, and poly_set_coeff.

Referenced by keypair().

5.26 precomp.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "precomp.h"
```

Include dependency graph for precomp.c:



Data Structures

- struct **tnode**
- struct **lnode**

Defines

- #define **INFINITY** (1.0 / 0.0)
- #define **EPSILON** (1.0 / (1UL << PREC_INTER))
- #define **ABOVE_MIN**(len, delta, z) ((len) + PREC_INTER - log2(delta) + (z) + EPSILON >= **min**)
- #define **MIN**(x, y) ((x < y) ? x : y)

Typedefs

- typedef struct **tnode** * **tree_t**
- typedef struct **lnode** * **list_t**

Functions

- **list_t list_alloc** (**tree_t** a, **list_t** s)
- **tree_t leaf_alloc** (int m, int t)
- **tree_t tree_alloc** (int m, int t, **tree_t** *s)
- int **l2** (unsigned long x)
- int **is_leaf** (int m, int t)
- double **bino_d** (int a, int b)
- double **binomial_d** (int a, int b)
- double **log_bino_d** (int a, int b)

- double **log_binomial_d** (int a, int b)
- double **dicho_si_lb_node** (int i, **distrib_t** d)
- double **dicho_si_lb_leaf** (int m, int t, int l)
- void **dicho_si_lb_rec** (int m, int t, **precomp_t** p, double **pe)
- double ** **dicho_si_lb** (**precomp_t** p)
- double **dicho_si_node** (int i, **distrib_t** d)
- double **dicho_si_leaf** (int m, int t, int l)
- void **dicho_si_rec** (int m, int t, **precomp_t** p, double **pe)
- double ** **dicho_si** (**precomp_t** p)
- double * **dicho_self_info_bounds** (**precomp_t** p)
- unsigned long **update_delta** (int i, **distrib_t** d, unsigned long delta)
- unsigned long **adjust_delta** (unsigned long delta, int *l)
- double **dicho_si_from_list** (**list_t** l, unsigned long delta, int len, double z, **precomp_t** p)
- double **tree_search** (**tree_t** a, **list_t** path, **list_t** todo, unsigned long delta, int len, double z, **precomp_t** p)
- **tree_t dicho_build_tree** (int m, int t, **precomp_t** p)
- void **clear_tree** (**tree_t** a, **precomp_t** p)
- double **dicho_searchmin** (**precomp_t** p, double min_value)
- **distrib_t init_proba** (int m, int t, int i)
- **leaf_info_t leaf_info** (int m, int t)
- double **max_si_loss** (int m, int t, **distrib_t** d)
- void **distrib_clear** (**distrib_t** dist)
- **precomp_t precomp_build** (int m, int t, int reduc)
- void **write_precomp** (**precomp_t** p, FILE *output_stream)
- void **clear_precomp** (**precomp_t** p)

Variables

- double ** **si_lb**
- double **min** = 0

5.26.1 Define Documentation

5.26.1.1 #define ABOVE_MIN(len, delta, z) ((len) + PREC_INTER - log2(delta) + (z) + EPSILON >= min)

Definition at line 314 of file precomp.c.

Referenced by dicho_si_from_list(), and tree_search().

5.26.1.2 #define EPSILON (1.0 / (1UL << PREC_INTER))

Definition at line 274 of file precomp.c.

Referenced by dicho_self_info_bounds().

5.26.1.3 #define INFINITY (1.0 / 0.0)

Definition at line 7 of file precomp.c.

Referenced by dicho_si_from_list(), log_binomial_d(), and tree_search().

5.26.1.4 #define MIN(x, y) ((x < y) ? x : y)

Referenced by clear_precomp(), precomp_build(), and write_precomp().

5.26.2 Typedef Documentation

5.26.2.1 typedef struct lnode * list_t

5.26.2.2 typedef struct tnode * tree_t

5.26.3 Function Documentation

5.26.3.1 unsigned long adjust_delta (unsigned long *delta*, int * *l*)

Definition at line 309 of file precomp.c.

References l2(), and PREC_INTER.

Referenced by dicho_si_from_list(), and tree_search().

5.26.3.2 double bino_d (int *a*, int *b*)

Definition at line 99 of file precomp.c.

Referenced by binomial_d().

5.26.3.3 double binomial_d (int *a*, int *b*)

Definition at line 105 of file precomp.c.

References bino_d().

Referenced by dicho_si_from_list(), dicho_si_lb_leaf(), dicho_si_leaf(), init_proba(), leaf_info(), main(), and max_si_loss().

5.26.3.4 void clear_precomp (precomp_t *p*)

Definition at line 702 of file precomp.c.

References precomp→distrib, distrib_clear(), is_leaf(), precomp→m, distrib_t→max, MIN, precomp→offset, and precomp→t.

Referenced by main().

5.26.3.5 void clear_tree (tree_t *a*, precomp_t *p*)

Definition at line 426 of file precomp.c.

References tnode→m, distrib_t→max, distrib_t→min, precomp_get_distrib, tnode→sons, and tnode→t.

Referenced by dicho_searchmin().

5.26.3.6 tree_t dicho_build_tree (int *m*, int *t*, precomp_t *p*)

Definition at line 400 of file precomp.c.

References is_leaf(), leaf_alloc(), distrib_t→max, distrib_t→min, precomp_get_distrib, and tree_alloc().

Referenced by dicho_searchmin().

5.26.3.7 double dicho_searchmin (precomp_t *p*, double *min_value*)

Definition at line 439 of file precomp.c.

References clear_tree(), dicho_build_tree(), dicho_si_lb(), precomp→m, min, PREC_INTER, precomp→real_m, precomp→real_t, si_lb, precomp→t, and tree_search().

Referenced by main().

5.26.3.8 double* dicho_self_info_bounds (precomp_t *p*)

Definition at line 276 of file precomp.c.

References dicho_si(), dicho_si_lb(), EPSILON, precomp→m, precomp→real_m, precomp→real_t, si_lb, and precomp→t.

Referenced by main().

5.26.3.9 double** dicho_si (precomp_t *p*)

Definition at line 254 of file precomp.c.

References dicho_si_rec(), precomp→m, and precomp→t.

Referenced by dicho_self_info_bounds().

5.26.3.10 double dicho_si_from_list (list_t *l*, unsigned long *delta*, int *len*, double *z*, precomp_t *p*)

Definition at line 316 of file precomp.c.

References ABOVE_MIN, adjust_delta(), binomial_d(), leaf_info_t→deadbits, INFINITY, precomp→leaf_info, tnode→m, min, lnode→next, PREC_INTER, si_lb, tnode→t, and lnode→tree.

Referenced by tree_search().

5.26.3.11 double** dicho_si_lb (precomp_t *p*)

Definition at line 188 of file precomp.c.

References dicho_si_lb_rec(), precomp→m, and precomp→t.

Referenced by dicho_searchmin(), and dicho_self_info_bounds().

5.26.3.12 double dicho_si_lb_leaf (int *m*, int *t*, int *l*)

Definition at line 147 of file precomp.c.

References binomial_d(), and PREC_INTER.

Referenced by dicho_si_lb_rec().

5.26.3.13 double dicho_si_lb_node (int *i*, distrib_t *d*)

Definition at line 134 of file precomp.c.

References distrib_get_proba, distrib_t→max, PREC_INTER, and PREC_PROBA.

Referenced by dicho_si_lb_rec(), and max_si_loss().

5.26.3.14 void dicho_si_lb_rec (int *m*, int *t*, precomp_t *p*, double ** *pe*)

Definition at line 159 of file precomp.c.

References leaf_info_t→deadbits, dicho_si_lb_leaf(), dicho_si_lb_node(), is_leaf(), precomp→leaf_info, distrib_t→max, distrib_t→min, and precomp_get_distrib.

Referenced by dicho_si_lb(), and dicho_si_rec().

5.26.3.15 double dicho_si_leaf (int *m*, int *t*, int *l*)

Definition at line 219 of file precomp.c.

References binomial_d().

Referenced by dicho_si_rec().

5.26.3.16 double dicho_si_node (int *i*, distrib_t *d*)

Definition at line 206 of file precomp.c.

References distrib_get_proba, distrib_t→max, and PREC_PROBA.

Referenced by dicho_si_rec().

5.26.3.17 void dicho_si_rec (int *m*, int *t*, precomp_t *p*, double ** *pe*)

Definition at line 227 of file precomp.c.

References leaf_info_t→deadbits, dicho_si_lb_rec(), dicho_si_leaf(), dicho_si_node(), is_leaf(), precomp→leaf_info, distrib_t→max, distrib_t→min, and precomp_get_distrib.

Referenced by dicho_si().

5.26.3.18 void distrib_clear (distrib_t *dist*)

Definition at line 524 of file precomp.c.

References distrib_t→prob.

Referenced by clear_precomp(), and precomp_build().

5.26.3.19 distrib_t init_proba (int *m*, int *t*, int *i*)

Definition at line 455 of file precomp.c.

References binomial_d(), distrib_get_proba, distrib_t→max, distrib_t→min, PREC_PROBA, distrib_t→prob, and round().

Referenced by precomp_build().

5.26.3.20 int is_leaf (int *m*, int *t*)

Definition at line 87 of file precomp.c.

5.26.3.21 int l2 (unsigned long *x*)

Definition at line 43 of file precomp.c.

5.26.3.22 tree_t leaf_alloc (int *m*, int *t*)

Definition at line 27 of file precomp.c.

References tnode→m, tnode→sons, and tnode→t.

Referenced by dicho_build_tree().

5.26.3.23 leaf_info_t leaf_info (int *m*, int *t*)

Definition at line 479 of file precomp.c.

References binomial_d(), leaf_info_t→deadbits, l2(), leaf_info_t→maximum, PREC_INTER, and PREC_PROBA.

Referenced by precomp_build().

5.26.3.24 list_t list_alloc (tree_t *a*, list_t *s*)

Definition at line 20 of file precomp.c.

References lnode→next, and lnode→tree.

Referenced by tree_search().

5.26.3.25 double log_bino_d (int *a*, int *b*)

Definition at line 113 of file precomp.c.

Referenced by log_binomial_d().

5.26.3.26 double log_binomial_d (int *a*, int *b*)

Definition at line 119 of file precomp.c.

References INFINITY, and log_bino_d().

Referenced by main().

5.26.3.27 double max_si_loss (int *m*, int *t*, distrib_t *d*)

Definition at line 508 of file precomp.c.

References binomial_d(), dicho_si_lb_node(), distrib_t→max, and distrib_t→min.

Referenced by precomp_build().

5.26.3.28 precomp_t precomp_build (int *m*, int *t*, int *reduc*)

Definition at line 528 of file precomp.c.

References precomp→distrib, distrib_clear(), init_proba(), is_leaf(), leaf_info(), precomp→leaf_info, precomp→m, distrib_t→max, max_si_loss(), MIN, distrib_t→min, precomp→offset, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by main().

5.26.3.29 tree_t tree_alloc (int *m*, int *t*, tree_t * *s*)

Definition at line 35 of file precomp.c.

References tnode→m, tnode→sons, and tnode→t.

Referenced by dicho_build_tree().

5.26.3.30 double tree_search (tree_t *a*, list_t *path*, list_t *todo*, unsigned long *delta*, int *len*, double *z*, precomp_t *p*)

Definition at line 360 of file precomp.c.

References ABOVE_MIN, adjust_delta(), dicho_si_from_list(), INFINITY, list_alloc(), tnode→m, distrib_t→max, distrib_t→min, lnode→next, precomp_get_distrib, si_lb, tnode→sons, tnode→t, lnode→tree, and update_delta().

Referenced by dicho_searchmin().

5.26.3.31 unsigned long update_delta (int *i*, distrib_t *d*, unsigned long *delta*)

Definition at line 300 of file precomp.c.

References distrib_get_proba, distrib_t→max, and PREC_PROBA.

Referenced by tree_search().

5.26.3.32 void write_precomp (precomp_t *p*, FILE * *output_stream*)

Definition at line 611 of file precomp.c.

References leaf_info_t→deadbits, precomp→distrib, distrib_get_proba, is_leaf(), precomp→leaf_info, precomp→m, distrib_t→max, leaf_info_t→maximum, MIN, distrib_t→min, min, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by main().

5.26.4 Variable Documentation

5.26.4.1 double min = 0

Definition at line 269 of file precomp.c.

Referenced by best_wf(), dicho_searchmin(), dicho_si_from_list(), main(), workfactor(), and write_precomp().

5.26.4.2 double** si_lb

Definition at line 268 of file precomp.c.

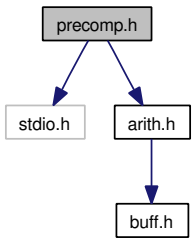
Referenced by dicho_searchmin(), dicho_self_info_bounds(), dicho_si_from_list(), and tree_search().

5.27 precomp.h File Reference

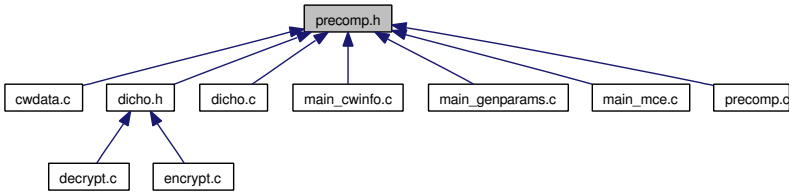
```
#include <stdio.h>
```

```
#include "arith.h"
```

Include dependency graph for precomp.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **leaf_info_t**
- struct **precomp**

Defines

- #define **precomp_get_distrib**(p, m, t) ((p).distrib[m][(t) - (p).offset[m]])

Typedefs

- typedef struct **precomp precomp_t**

Functions

- double **binomial_d** (int a, int b)
- double **log_binomial_d** (int a, int b)
- void **clear_precomp** (**precomp_t** p)
- void **write_precomp** (**precomp_t** p, FILE *output_stream)
- **precomp_t precomp_build** (int m, int t, int reduc)
- double **dicho_searchmin** (**precomp_t** p, double min_value)
- double * **dicho_self_info_bounds** (**precomp_t** p)

5.27.1 Define Documentation

5.27.1.1 #define precomp_get_distrib(p, m, t) ((p).distrib[m][(t) - (p).offset[m]])

Definition at line 18 of file precomp.h.

Referenced by clear_tree(), dicho_build_tree(), dicho_rec(), dicho_si_lb_rec(), dicho_si_rec(), dichoinv_rec(), and tree_search().

5.27.2 Typedef Documentation

5.27.2.1 typedef struct precomp precomp_t

5.27.3 Function Documentation

5.27.3.1 double binomial_d (int *a*, int *b*)

Definition at line 105 of file precomp.c.

References bino_d().

Referenced by dicho_si_from_list(), dicho_si_lb_leaf(), dicho_si_leaf(), init_proba(), leaf_info(), main(), and max_si_loss().

5.27.3.2 void clear_precomp (precomp_t *p*)

Definition at line 702 of file precomp.c.

References precomp→distrib, distrib_clear(), is_leaf(), precomp→m, distrib_t→max, MIN, precomp→offset, and precomp→t.

Referenced by main().

5.27.3.3 double dicho_searchmin (precomp_t *p*, double *min_value*)

Definition at line 439 of file precomp.c.

References clear_tree(), dicho_build_tree(), dicho_si_lb(), precomp→m, min, PREC_INTER, precomp→real_m, precomp→real_t, si_lb, precomp→t, and tree_search().

Referenced by main().

5.27.3.4 double* dicho_self_info_bounds (precomp_t *p*)

Definition at line 276 of file precomp.c.

References dicho_si(), dicho_si_lb(), EPSILON, precomp→m, precomp→real_m, precomp→real_t, si_lb, and precomp→t.

Referenced by main().

5.27.3.5 double log_binomial_d (int *a*, int *b*)

Definition at line 119 of file precomp.c.

References INFINITY, and log_bino_d().

Referenced by main().

5.27.3.6 precomp_t precomp_build (int *m*, int *t*, int *reduc*)

Definition at line 528 of file precomp.c.

References precomp→distrib, distrib_clear(), init_proba(), is_leaf(), leaf_info(), precomp→leaf_info, precomp→m, distrib_t→max, max_si_loss(), MIN, distrib_t→min, precomp→offset, precomp→real_m, precomp→real_t, and precomp→t.

Referenced by main().

5.27.3.7 void write_precomp (precomp_t *p*, FILE * *output_stream*)

Definition at line 611 of file precomp.c.

References leaf_info_t→deadbits, precomp→distrib, distrib_get_proba, is_leaf(), precomp→leaf_info, precomp→m, distrib_t→max, leaf_info_t→maximum, MIN, distrib_t→min, min, precomp→real_m, precomp→real_t, and precomp→t.

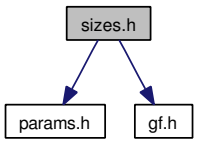
Referenced by main().

5.28 sizes.h File Reference

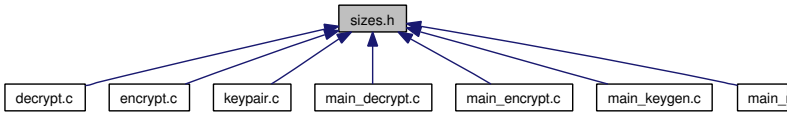
```
#include "params.h"
```

```
#include "gf.h"
```

Include dependency graph for sizes.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define **NB_ERRORS** ERROR_WEIGHT
- #define **EXT_DEGREE** LOG_LENGTH
- #define **LENGTH** (1 << EXT_DEGREE)
- #define **CODIMENSION** (NB_ERRORS * EXT_DEGREE)
- #define **DIMENSION** (LENGTH - CODIMENSION)
- #define **BITS_TO_BYTES**(nb_bits) (((nb_bits) - 1) / 8 + 1)
- #define **BIT_SIZE_OF_LONG** (8 * sizeof(long))
- #define **BITS_TO_LONG**(nb_bits) (((nb_bits) - 1) / BIT_SIZE_OF_LONG + 1)
- #define **SECRETKEY_BYTES** (LENGTH * sizeof (long) * BITS_TO_LONG(CODIMENSION) + (LENGTH + 1 + (NB_ERRORS + 1) * NB_ERRORS) * sizeof (**gf_t**))
- #define **PUBLICKEY_BYTES** (BITS_TO_LONG(CODIMENSION) * sizeof(long) * DIMENSION)
- #define **CLEARTEXT_LENGTH** (DIMENSION + ERROR_SIZE)
- #define **CLEARTEXT_BYTES** BITS_TO_BYTES(CLEARTEXT_LENGTH)
- #define **CIPHERTEXT_BYTES** BITS_TO_BYTES(LENGTH)

5.28.1 Define Documentation

5.28.1.1 #define BIT_SIZE_OF_LONG (8 * sizeof(long))

Definition at line 14 of file sizes.h.

Referenced by keypair(), syndrome(), and vec_concat().

5.28.1.2 #define BITS_TO_BYTES(nb_bits) (((nb_bits) - 1) / 8 + 1)

Definition at line 12 of file sizes.h.

Referenced by decrypt_block(), and vec_concat().

5.28.1.3 #define BITS_TO_LONG(nb_bits) (((nb_bits) - 1) / BIT_SIZE_OF_LONG + 1)

Definition at line 16 of file sizes.h.

Referenced by addto(), encrypt_block(), keypair(), sk_from_string(), syndrome(), and xor().

5.28.1.4 #define CIPHERTEXT_BYTES BITS_TO_BYTES(LENGTH)

Definition at line 25 of file sizes.h.

Referenced by main().

5.28.1.5 #define CLEARTEXT_BYTES BITS_TO_BYTES(CLEARTEXT_LENGTH)

Definition at line 23 of file sizes.h.

Referenced by check(), and main().

5.28.1.6 #define CLEARTEXT_LENGTH (DIMENSION + ERROR_SIZE)

Definition at line 21 of file sizes.h.

Referenced by check(), and main().

5.28.1.7 #define CODIMENSION (NB_ERRORS * EXT_DEGREE)

Definition at line 8 of file sizes.h.

Referenced by addto(), encrypt_block(), keypair(), sk_from_string(), syndrome(), vec_concat(), and xor().

5.28.1.8 #define DIMENSION (LENGTH - CODIMENSION)

Definition at line 9 of file sizes.h.

Referenced by decrypt_block(), encrypt_block(), and vec_concat().

5.28.1.9 #define EXT_DEGREE LOG_LENGTH

Definition at line 5 of file sizes.h.

Referenced by decode(), key_genmat(), keypair(), main(), roots_berl(), roots_berl_aux(), and syndrome().

5.28.1.10 #define LENGTH (1 << EXT_DEGREE)

Definition at line 7 of file sizes.h.

Referenced by key_genmat(), keypair(), sk_from_string(), and syndrome().

5.28.1.11 #define NB_ERRORS ERROR_WEIGHT

Definition at line 4 of file sizes.h.

Referenced by decode(), decrypt_block(), encrypt_block(), key_genmat(), keypair(), main(), roots_berl(), roots_berl_aux(), sk_free(), sk_from_string(), and syndrome().

5.28.1.12 #define PUBLICKEY_BYTES (BITS_TO_LONG(CODIMENSION) * sizeof(long) * DIMENSION)

Definition at line 19 of file sizes.h.

Referenced by main().

5.28.1.13 #define SECRETKEY_BYTES (LENGTH * sizeof (long) * BITS_TO_LONG(CODIMENSION) + (LENGTH + 1 + (NB_ERRORS + 1) * NB_ERRORS) * sizeof (gf_t))

Definition at line 18 of file sizes.h.

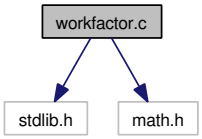
Referenced by main().

5.29 workfactor.c File Reference

`#include <stdlib.h>`

`#include <math.h>`

Include dependency graph for workfactor.c:



Functions

- double **binomial** (int n, int k)
- double **log_binomial** (int n, int k)
- double **nb_iter** (int n, int k, int w, int p, int l)
- double **cout_iter** (int n, int k, int p, int l)
- double **memory_compl** (int n, int k, int p, int l)
- double **cout_total** (int n, int k, int w, int p, int l)
- double **best_wf** (int n, int k, int w, int p, int *lmin, double *mem)
- double **workfactor** (int n, int k, int t)

5.29.1 Function Documentation

5.29.1.1 double best_wf (int *n*, int *k*, int *w*, int *p*, int * *lmin*, double * *mem*)

Definition at line 76 of file workfactor.c.

References cout_total(), memory_compl(), and min.

Referenced by workfactor().

5.29.1.2 double binomial (int *n*, int *k*)

Definition at line 4 of file workfactor.c.

Referenced by cout_iter(), and memory_compl().

5.29.1.3 double cout_iter (int *n*, int *k*, int *p*, int *l*)

Definition at line 38 of file workfactor.c.

References binomial().

Referenced by cout_total().

5.29.1.4 double cout_total (int *n*, int *k*, int *w*, int *p*, int *l*)

Definition at line 68 of file workfactor.c.

References cout_iter(), and nb_iter().

Referenced by best_wf(), and workfactor().

5.29.1.5 double log_binomial (int n, int k)

Definition at line 16 of file workfactor.c.

Referenced by nb_iter().

5.29.1.6 double memory_compl (int n, int k, int p, int l)

Definition at line 58 of file workfactor.c.

References aux, and binomial().

Referenced by best_wf().

5.29.1.7 double nb_iter (int n, int k, int w, int p, int l)

Definition at line 28 of file workfactor.c.

References log_binomial().

Referenced by cout_total().

5.29.1.8 double workfactor (int n, int k, int t)

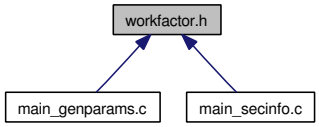
Definition at line 117 of file workfactor.c.

References best_wf(), cout_total(), and min.

Referenced by main().

5.30 workfactor.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- double **workfactor** (int n, int k, int t)

5.30.1 Function Documentation

5.30.1.1 double workfactor (int *n*, int *k*, int *t*)

Definition at line 117 of file workfactor.c.

References best_wf(), cout_total(), and min.

Referenced by main().