Enunciado

1. Usando o protocolo OT construído na questão anterior
   A. Implemente o protocolo sVOLE (“subset vectorial oblivious linear evaluation”)
   B. Usando sVOLE implemente um protótipo de um protocolo ZK-sVOLE usando equações polinomiais do 2º grau aleatoriamente geradas.

PARTE A

Definição de variáveis

Para definir o protocolo vamos começar por fixar os parâmetros, p, k e ℓ, que determinam a estrutura algébrica onde o protocolo se desenvolve.

In [1]:

```
p = 3
k = 5
l = 8
lambda_security = 128
N = 2 * k

print_messagens_geradas = True
```

Estruturas algébricas

In [2]:

```
F = GF(p)
R.<x> = PolynomialRing(F)
f = R.irreducible_element(k)
E.<a> = GF(p^k, name='a', modulus=f)

def verifica_base(Z, E):
    F = GF(E.characteristic())
    V = VectorSpace(F, E.degree())
    for j in range(len(Z)):
        subZ = Z[:j] + Z[j+1:]
        mat = matrix(F, [V(E(z)) for z in subZ])
        if mat.rank() < E.degree():
            return False
    return True

def gerarZ(E, N):
    F = GF(E.characteristic())
    k = E.degree()
    V = VectorSpace(F, k)
    tentativas = 0
    while True:
        tentativas += 1
        Z = set()
        while len(Z) < N:
            v = V.random_element()
            if not v.is_zero():
                Z.add(E(v))
        Z = list(Z)
        if verifica_base(Z, E):
            print(f"Conjunto Z encontrado após {tentativas} tentativas.")
            return Z

Z = gerarZ(E,N)
```

```
Conjunto Z encontrado após 1 tentativas.
```

Esta estrutura algébrica fornece a base para o protocolo OLE, que permite a avaliação linear de forma “oblivious”, sendo uma primitiva fundamental em construção de protocolos criptográficos como OT.

Oblivious transfer da questão anterior:

In [3]:

```
import random
import hashlib

def bernoulli(epsilon, n=53):
    """
    Gera uma amostra de Bernoulli B(epsilon) usando a construção de Lebesgue.
    - epsilon: parâmetro da distribuição de Bernoulli (0 < epsilon < 1)
    - n: número de bits para a precisão (default: 53, precisão de um double)
    """
    # Gera a string de bits aleatórios {0,1}^n
    w = [random.randint(0, 1) for _ in range(1, n+1)]

    # Calcula o racional de Lebesgue
    w_hat = sum(w[i-1] * 2^(-i) for i in range(1,n+1))

    return 1 if w_hat <= epsilon else 0

def bernoulli_lambda(epsilon, n=53, lambda_ = lambda_security):
    return [bernoulli(epsilon,n) for _ in range(lambda_)]

def vector_to_bytes(vector, p):
    byte_length = (p.bit_length() + 7) // 8
    return b''.join(
        int(x).to_bytes(byte_length, 'big', signed=False)
    for x in vector)

def bytes_to_vector(byte_stream, p, l):
    byte_length = (p.bit_length() + 7) // 8
    elementos = []
    for i in range(0, len(byte_stream), byte_length):
        chunk = byte_stream[i:i+byte_length]
        if not chunk:
            elementos.append(F(0))
            continue
        valor = int.from_bytes(chunk, 'big') % p
        elementos.append(F(valor))
    return vector(F, elementos[:l])

def converter_bytes_para_mensagens(byte_stream_list, p, tamanho, N):
    byte_length = (p.bit_length() + 7) // 8
    mensagens = {}
    for n in range(1, N + 1):
        if n-1 < len(byte_stream_list) and byte_stream_list[n-1] is not None:
            byte_stream = byte_stream_list[n-1]
            elementos = []
            for i in range(0, len(byte_stream), byte_length):
                chunk = byte_stream[i:i+byte_length]
                if not chunk:
                    elementos.append(F(0))
                    continue
                valor = int.from_bytes(chunk, 'big') % p
                elementos.append(F(valor))
            # Preenche com zeros se faltarem elementos
            while len(elementos) < tamanho:
                elementos.append(F(0))
            mensagens[n] = vector(F, elementos[:tamanho])
        else:
            mensagens[n] = vector(F, [F(0)] * tamanho)
    return mensagens

def xof(seed: bytes, nbytes: int):
    shake = hashlib.shake_128()
    shake.update(seed)
    return shake.digest(nbytes)

def xof_bits(seed: bytes, nbits: int):
    nbytes = (nbits + 7) // 8
    output = xof(seed, nbytes)
    bits = []
    for byte in output:
        for i in range(8):
            bits.append((byte >> (7 - i)) & 1)
            if len(bits) == nbits:
                return bits
```

In [4]:

```
class Sender:
    def __init__(self, alpha, ell, xof_name="shake128"):
        self.alpha = alpha
        self.l = ell
        self.xof_name = xof_name
        self.seed = f"{alpha}:{lambda_security}".encode()
        self.criterion = None
        self.pks = []
        self.N = N
        self.mensagens = None
        self.r = []
        self.criptogramas = []
        self.t = 100

    def get_vector_ai_ui(self, i, lambda_):
        seed_i = self.seed + f":{i+1}".encode()
        bits = xof_bits(seed_i, lambda_ + 1)
        a_i = bits[:lambda_]
        u_i = bits[lambda_]
        return a_i, u_i

    def get_criterion_sequence(self, lambda_):
        self.criterion = [self.get_vector_ai_ui(i, lambda_) for i in range(self.l)]

    def receive_and_verifypks(self, pks):
        self.pks = pks
        for i in range(self.l):
            soma = sum(pks[i][klinha] for klinha in range(self.N)) % 2
            _, u_i = self.criterion[i]
            if soma != u_i:
                print(f"FALHA na linha i={i}: soma={soma}, u_i={u_i}")
                print(f"pks[{i}] = {pks[i]}")
                print(f"u_{i} = {u_i}")
                print("------------------")
                return False
        print("Verificação OK: todas as somas coincidem com u_i")
        print("------------------")
        return True

    def gerar_r(self, delta, p=0.1):
        while True:
            r = [bernoulli(p) for _ in range(self.l)]
            if sum(r) <= delta:
                self.r = r
                return

    def calcular_criptogramas(self):
        self.criptogramas = []
        for _ in range(self.t):
            self.gerar_r(128)
            a = [0] * lambda_security  # Agora em F_p
            for i in range(self.l):
                if self.r[i] == 1:
                    a_i, _ = self.criterion[i]
                    a = [(a[j] + a_i[j]) % p for j in range(lambda_security)]
            criptogramas_k = []
            for k in range(1,N+1):
                msg_vector = self.mensagens[k]  # msg_vector é um vetor em F_p^ℓ
                c_k = []
                for elemento in msg_vector:
                    soma = sum(self.r[i] * self.pks[i][k-1] for i in range(self.l)) % 2
                    c_k.append((elemento + soma) % p)
                criptogramas_k.append(c_k)
            self.criptogramas.append((a, criptogramas_k))

    def print_info(self):
        print("self.alpha: ",self.alpha)
        print("self.l: ",self.l)
        print("self.xof_name: ",self.xof_name)
        print("self.seed: ",self.seed)
        print("self.criterion: ",self.criterion)
```

In [5]:

```
class Receiver:
    def __init__(self, eps, b, lambda_, n_mensagens):
        self.alpha = None
        self.l = None
        self.eps = eps
        self.secrets = []
        self.criterion = None
```

```
        self.N = n_mensagens

        self.t = None
        self.b = b - 1

    def receive_alpha_l(self, alpha, l):
        self.alpha = alpha
        self.l = l
        self.seed = f"{alpha}:{lambda_security}".encode()
        self.criterion = self.get_criterion_sequence(lambda_security)

    def get_vector_ai_ui(self, i, lambda_):
        seed_i = self.seed + f":{i+1}".encode()
        bits = xof_bits(seed_i, lambda_ + 1)
        a_i = bits[:lambda_]
        u_i = bits[lambda_]
        return a_i, u_i

    def get_criterion_sequence(self, lambda_):
        if self.l is None:
            raise ValueError("self.l não está definido")
        #print(f"Seed usado: {self.seed}")
        criterion = [self.get_vector_ai_ui(i, lambda_) for i in range(self.l)]
        return criterion

    def generate_N_secrets(self):
        self.secrets = []
        for k in range(self.N):
            if k == self.b:
                self.secrets.append(None)  # s_b = ⊥
            else:
                s_k = bernoulli_lambda(self.eps, lambda_security)
                self.secrets.append(s_k)

    def generate_pks(self):
        self.t = [[0 for _ in range(self.N)] for _ in range(self.l)]
        for i in range(self.l):
            a_i, u_i = self.criterion[i]
            soma = 0
            for k in range(self.N):
                if k != self.b:
                    s_k = self.secrets[k]
                    dot = sum([a_i[j] * s_k[j] for j in range(lambda_security)]) % 2
                    e = bernoulli(self.eps)
                    t_ik = (dot + e) % 2
                    self.t[i][k] = t_ik
                    soma = (soma + t_ik) % 2
                else:
                    self.t[i][k] = None
            self.t[i][self.b] = (u_i - soma) % 2
        return self.t

    def recuperar_mensagens(self, a, c):
        mensagens_recuperadas = []
        for k in range(N):
            if k == self.b:
                mensagens_recuperadas.append(None)
            else:
                s_k = self.secrets[k]
                # Garantir que a e s_k estão no mesmo campo
                a_dot_s = sum(F(a[j]) * F(s_k[j]) for j in range(lambda_security)) % p
                msg_recuperada = [F((c[k][i] + a_dot_s) % p) for i in range(len(c[k]))]
                mensagens_recuperadas.append(msg_recuperada)
        return mensagens_recuperadas

    def recuperar_mensagens_maioritarias(self, lista_criptogramas):
        t = len(lista_criptogramas)
        resultados_por_k = [[] for _ in range(self.N)]
        for i in range(t):
            a, c = lista_criptogramas[i]
            mks = self.recuperar_mensagens(a, c)
            for k in range(self.N):
                resultados_por_k[k].append(tuple(mks[k]) if mks[k] is not None else None)
        mensagens_finais = []
        for k in range(self.N):
            if k == self.b:
                mensagens_finais.append(None)
            else:
                contagem = {}
                for msg in resultados_por_k[k]:
                    if msg is not None:
                        # Ignora vetores totalmente zero
                        if all(x == 0 for x in msg):
                            continue
```

```
                    msg_key = tuple(msg)

                    contagem[msg_key] = contagem.get(msg_key, 0) + 1
            if contagem:
                mensagem_mais_votada = max(contagem.items(), key=lambda x: x[1])[0]
                byte_msg = vector_to_bytes(mensagem_mais_votada, p)
            else:
                byte_msg = None  # Todos eram vetores de zeros
            mensagens_finais.append(byte_msg)
        return mensagens_finais
```

Protocolo sVOLE

O protocolo OLE simples tem simultaneamente uma questão de segurança e uma questão de eficiência ambas com a mesma proveniência: o número de elementos p^k do corpo E.

- A questão de segurança resulta de, para existir capacidade de verificar o conhecimento de $x \in E$ ambos os "tags" m, $\mathsf{q} \in E$ têm de ser suficientemente grandes. Assim o tamanho dessas "tags", que no protocolo OLE é $|p| \times k$, exige um valor mínimo razoável (~ 128 bits, ou superior).
- Ainda, em termos de segurança, o tamanho da chave global Δ tem de ser sufientemente grande para proteger o protocolo de ataques. Como Δ é um valor gerado de E o factor "segurança de Δ" exige também um valor mínimo para p^k.
- A questão de eficiência resulta do facto de o número N de mensagens usadas no protocolo OT e usadas nas somas que calculam os "tags" ser também p^k. A eficiência exige que N não possa ser muito grande.

Para criar uma solução que responda simultaneamente a todas estas questões, a partir de um factor de segurança λ e vamos seguir os seguintes objectivos

1. Os "tags" m e q passam a ser vetores com ℓ componentes em E; como elementos de E^ℓ o o seu tamanho é substancialmente aumentado para $|p| \times k \times \ell$; se a dimensão ℓ for suficientemente grande, mesmo com $|p|$ e k relativamente pequenos
2. Os índices z que identificam os elementos t_z usados no protocolo OT formam um subconjunto $\mathcal{Z} \equiv \{z_1, z_2, \cdots z_N\} \subset E$ com $N = \text{poly}(\lambda)$ elementos. O valor de N e os elementos $\{z_n\}_{n \in [N]}$ são escolhidos de tal forma que, para todo $n \in [N]$ o conjunto $\mathcal{Z} \setminus \{z_n\}$ é uma base de F_{p^k} visto como espaço vectorial sobre F_p.
   Concretamente, cada $a \in E$ é uma combinação linear, com coeficientes em F, dos elementos de $\mathcal{Z}$ e , cada $z_n \in \mathcal{Z}$ é uma combinação linear dos elementos de $\mathcal{Z} \setminus \{z_n\}$.
3. Os valores a autenticar $\underline{x} = \langle x_1, x_2, \cdots, x_\ell \rangle$ passam a ser também vectores de domensão ℓ; isto é $\underline{x} \in F^\ell$.
4. A chave global Δ é escolhida, em cada instância do protocolo, dentro do conjunto $\mathcal{Z}$ através da seleção de um índice $j \leftarrow [N]$, e depois escolhendo $\Delta \equiv z_j$.

   $\Delta\,\gets\,\{\,\vartheta\,j\gets [N]\,\centerdot\, z_j\,\}$

5. Os valores $t_z \in F$ do protocolo OLE são substituídos por vectores; assim, para todo $i \in [\ell]$, define-se um vector $\underline{t_i} = \{t_{i,n}\}_{n \in [N]} \in F^\ell$ em que,

   1. para $\,n>1\,$, tem-se $\,t_{i,n}\,\gets\,F\,$ é gerado pseudo-aleatoriamente
   2. para $\,n=1\,$ tem-se $t_{i,1}\,\gets\, x_i - \sum_{n=2}^N\,t_{i,n}$.

   Isto assegura que, para todo $i \in [\ell]$, se verifica $x_i = \sum_{n \in [N]} t_{i,n}$

1. Ambos os agentes, Prover e Verifier conhecem os parâmetros p, k, ℓ e o parâmetro de segurança λ. Conhecem também $\mathcal{Z} \subset \mathrm{F}_{p^k}$ e a sua cardinalidade $N = O(\text{poly}(\lambda))$, como especificámos atrás.

Prover sVOLE

Este agente conhece um vector $\underline{x} \in F^\ell$, que é sua informação privada, e procede do seguinte modo

a. para todo $i \in [\ell]$ \ i. Gera o vector $\underline{t_i}$ como está indicado no ponto 5. acima, \ ii. Disponibiliza N mensagens $\{m_{i,n} \leftarrow \sigma(t_{i,n})\}_{n \in [N]}$ para transferência num protocolo "oblivious transfer" $\binom{N}{N-1}$. \ iii. Calcula o "tag" $\mathsf{m}_i \leftarrow \sum_{n \in [N]} (-z_n) \cdot t_{i,n}$.

b. Agrega num vector $\underline{\mathsf{m}} \in E^\ell$ os vários "tags" $\{\mathsf{m}_i\}_{i \in [\ell]}$

In [6]:

```
class Prover_sVOLE:
    def __init__(self,ell):
        self.ell = ell
        self.vectorX = vector(F, [F.random_element() for _ in range(self.ell)])
        self.t_matrix = {}
        self.messages = []
        self.sender = Sender(alpha="alpha123", ell = self.ell)
        self.tag = None

    #passo 2 i
```

```
    #passo a.i
    def gerar_tz(self):
        for i in range(len(self.vectorX)):
            t_i = {}
            for n in range(2, N + 1):
                t_i[n] = F.random_element()
            t_i[1] = self.vectorX[i] - sum(t_i.values())
            self.t_matrix[i] = {k: t_i[k] for k in sorted(t_i.keys())}

    def mostrar_estrutura(self):
        print(f"Vetor privado x: {self.vectorX}")
        print("\nMatriz t_{i,n}:")
        for i in range(min(3, len(self.vectorX))):  # Mostra apenas as 3 primeiras linhas para exemplo
            print(f"i={i}: {self.t_matrix.get(i, 'Não gerado')}")

    #ponto a.ii
    def gen_messages(self):
        self.messages = {}
        for n in range(1, N + 1):
            msg_vector = [self.t_matrix[i][n] for i in range(self.ell)]
            byte_msg = vector_to_bytes(vector(F, msg_vector), p)
            self.messages[n] = byte_msg

        if print_messagens_geradas:
            print("Mensagens geradas:")
            for n in range(1, N + 1):
                print(f"{n}: {self.messages[n]}")
            print("------------------")

    # Função sigma que codifica um elemento de F como string de bits
    def _sigma(self, x):
        bit_length = (p - 1).bit_length()

        if x in F:
            return format(int(x), f'0{bit_length}b')
        elif isinstance(x, list) and all(y in F for y in x):
            return ''.join([format(int(y), f'0{bit_length}b') for y in x])
        else:
            raise ValueError("x deve ser um elemento de F ou uma lista de elementos de F")

    #ponto a.iii / b
    def gerar_tag(self):
        self.tag = []
        for i in range(len(self.vectorX)):
            soma = E(0)
            for n in range(1, N + 1):
                z_n = Z[n - 1]
                t_in = self.t_matrix[i][n]
                soma += (-z_n) * E(t_in)
            self.tag.append(soma)
        #print("Tag calculado:", self.tag)
        print("------------------")

prover_sVOLE = Prover_sVOLE(l)
prover_sVOLE.gerar_tz()
prover_sVOLE.mostrar_estrutura()
prover_sVOLE.gen_messages()

from sage.modules.vector_modn_dense import Vector_modn_dense

def dicionario_para_vetores(messages_dict, p, tamanho):
    byte_length = (p.bit_length() + 7) // 8
    vetores = {}
    for n in messages_dict:
        byte_stream = messages_dict[n]
        elementos = []
        for i in range(0, len(byte_stream), byte_length):
            chunk = byte_stream[i:i+byte_length]
            valor = int.from_bytes(chunk, 'big') % p
            elementos.append(F(valor))
        # Preenche com zeros se faltarem elementos
        while len(elementos) < tamanho:
            elementos.append(F(0))
        vetores[n] = vector(F, elementos[:tamanho])
    return vetores

# Uso:
vetores_OT = dicionario_para_vetores(prover_sVOLE.messages, p, l)
print("vetores_OT: ",vetores_OT)
prover_sVOLE.sender.mensagens = vetores_OT
prover_sVOLE.gerar_tag()
```

```
Vetor privado x: (2, 0, 0, 1, 0, 0, 1, 0)

Matriz t_{i,n}:
i=0: {1: 1, 2: 1, 3: 2, 4: 1, 5: 1, 6: 0, 7: 1, 8: 1, 9: 1, 10: 2}
i=1: {1: 2, 2: 0, 3: 1, 4: 1, 5: 1, 6: 0, 7: 0, 8: 0, 9: 0, 10: 1}
i=2: {1: 0, 2: 2, 3: 1, 4: 1, 5: 2, 6: 1, 7: 1, 8: 2, 9: 1, 10: 1}
Mensagens geradas:
1: b'\x01\x02\x00\x00\x01\x01\x01\x02'
2: b'\x01\x00\x02\x00\x02\x00\x00\x00'
3: b'\x02\x01\x01\x01\x00\x01\x02\x00'
4: b'\x01\x01\x01\x00\x00\x02\x01\x02'
5: b'\x01\x01\x02\x02\x01\x01\x00\x02'
6: b'\x00\x00\x01\x00\x01\x02\x02\x00'
7: b'\x01\x00\x01\x01\x02\x02\x00\x02'
8: b'\x01\x00\x02\x01\x02\x01\x01\x02'
9: b'\x01\x00\x01\x01\x01\x01\x00\x00'
10: b'\x02\x01\x01\x01\x02\x01\x00\x02'
------------------
vetores_OT:  {1: (1, 2, 0, 0, 1, 1, 1, 2), 2: (1, 0, 2, 0, 2, 0, 0, 0), 3: (2, 1, 1, 1, 0, 1, 2, 0),
4: (1, 1, 1, 0, 0, 2, 1, 2), 5: (1, 1, 2, 2, 1, 1, 0, 2), 6: (0, 0, 1, 0, 1, 2, 2, 0), 7: (1, 0, 1,
1, 2, 2, 0, 2), 8: (1, 0, 2, 1, 2, 1, 1, 2), 9: (1, 0, 1, 1, 1, 1, 0, 0), 10: (2, 1, 1, 1, 2, 1, 0,
2)}
------------------
```

Verifier sVOLE

Verifier

- para todo
  1. Gera $\Delta \leftarrow \{\vartheta j \leftarrow [N] \cdot z_j\}$ que passa a ser designada por chave global.
  2. para todo $i \in [\ell]$
     A. No i-ésimo protocolo $\binom{N}{N-1}$-OT inicializado pelo Prover, transfere as $N-1$ mensagens $\{m_{i,n}\}_{n \in [N] \setminus \{j\}}$
     B. Calcula $\mathbf{q}_i \leftarrow \sum_{n \in [N] \setminus \{j\}} (\Delta - z_n) \cdot \sigma^{-1}(m_{i,n})$
  3. Agrega num único vector $\mathbf{q} \in E^\ell$ todas as "tags" $\{\mathbf{q}_i\}_{i \in [\ell]}$

In [7]:

```python
class Verifier_sVOLE:
    def __init__(self,ell,jota,ene):
        self.ell = ell
        self.j = jota
        self.Delta = Z[self.j-1]
        self.q = []
        self.N = ene
        self.receiver = Receiver(0.1,self.j,lambda_security,self.N)

    # Passo 1: Escolhe Delta = z_j, onde z_j ∈ Z
    def gerar_delta(self):
        self.Delta = Z[self.j-1]
        print(f"Δ = {self.Delta} (índice j = {self.j})")
        print("------------------")

    # Passo 2.1: Obtém mensagens do Prover exceto a do índice j
    def participar_OT(self, mensagens):
        self.mensagens_OT = {n: mensagens.get(n, None) for n in range(1, N + 1)}
        """
        for n in mensagens:
            if n != self.j:
                print(f"m_{n}: {mensagens[n]}")
        print("------------------")
        """

    # Passo 2.2: Calcula q_i
    def calcular_q_i(self, i):
        soma = E(0)
        for n in range(1, self.N + 1):
            if n == self.j:
                continue
            mensagem = self.mensagens_OT.get(n, None)
            if mensagem is None:
                raise ValueError(f"Mensagem {n} não encontrada ou é None")
            z_n = Z[n - 1]
            m_in = mensagem[i]
            t_in = self._sigma_inversa(m_in)
            soma += (self.Delta - z_n) * E(t_in)
        return soma

    # Passo 3: Agrega todos q_i em q
```

```
    def calcular_q(self):

        self.q = [self.calcular_q_i(i) for i in range(self.ell)]
        #print("Vetor q¯:", self.q)
        print("------------------")

    # Decodifica mensagem (bits para elemento de F)
    def _sigma_inversa(self, bit_str):
        from sage.modules.vector_modn_dense import Vector_modn_dense

        # Se for string binária
        if isinstance(bit_str, str):
            return F(int(bit_str, 2))

        # Se for um vetor do Sage ou sequência
        if hasattr(bit_str, '__getitem__'):
            return F(bit_str[0])  # Pega o primeiro elemento

        # Se já for um elemento conversível
        return F(bit_str)

    def verificar_relacao(self, prover_tag, prover_x):
        resultados = []
        for i in range(self.ell):
            q_i = self.q[i]
            x_i = F(prover_x[i])
            m_i = prover_tag[i]

            x_i_E = E(x_i)
            delta_E = self.Delta  # Já está em E

            # Calcula x_i*Δ + m_i (tudo em E)
            lado_direito = x_i_E * delta_E + m_i

            print(f"i={i}: q_i = {q_i}")
            print(f"x_i*Δ + m_i = {x_i_E}*{delta_E} + {m_i} = {lado_direito}")
            print(f"Resultado: {q_i == lado_direito}")
            print("------------------")

            resultados.append(q_i == lado_direito)
        return all(resultados)
```

In [8]:

```
verifier_sVOLE = Verifier_sVOLE(l,random.randint(1, N),N)
verifier_sVOLE.gerar_delta()
prover_sVOLE.sender.get_criterion_sequence(lambda_security)
verifier_sVOLE.receiver.receive_alpha_l("alpha123", l)  # Corrigido para usar l=8
verifier_sVOLE.receiver.generate_N_secrets()
verifier_sVOLE.receiver.generate_pks()
prover_sVOLE.sender.receive_and_verifypks(verifier_sVOLE.receiver.t)
prover_sVOLE.sender.calcular_criptogramas()

mensagens_decodificadas = verifier_sVOLE.receiver.recuperar_mensagens_maioritarias(prover_sVOLE.sender.criptogram
as)
mensagens_decodificadas = converter_bytes_para_mensagens(mensagens_decodificadas, p, l, N)

for n in prover_sVOLE.sender.mensagens:
    if verifier_sVOLE.j != n:
        if prover_sVOLE.sender.mensagens[n] == mensagens_decodificadas[n]:
            print(f"Mensagem n={n} bem decifrada")
        else:
            print(f"Diferença em n={n}:")
            print(f"Original: {prover_sVOLE.sender.mensagens[n]}")
            print(f"Decodificado: {mensagens_decodificadas[n]}")
            print(f"Iguais? {prover_sVOLE.sender.mensagens[n] == mensagens_decodificadas[n]}")
    else:
        if mensagens_decodificadas[n] is None:
            print(f"Mensagem n={n} (índice b) foi corretamente omitida")

verifier_sVOLE.participar_OT(mensagens_decodificadas)
verifier_sVOLE.calcular_q()
resultado = verifier_sVOLE.verificar_relacao(prover_sVOLE.tag, prover_sVOLE.vectorX)
print("Verificação bem-sucedida?", resultado)
```

```
Δ = a^4 + 2*a^3 + a^2 + 2*a (índice j = 9)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem n=1 bem decifrada
Mensagem n=2 bem decifrada
Mensagem n=3 bem decifrada
Mensagem n=4 bem decifrada
Mensagem n=5 bem decifrada
Mensagem n=6 bem decifrada
Mensagem n=7 bem decifrada
Mensagem n=8 bem decifrada
Mensagem n=10 bem decifrada
------------------
i=0: q_i = 2*a^3 + 2*a^2 + 2*a + 1
x_i*Δ + m_i = 2*a^4 + 2*a^3 + a^2 + 2*a + a^4 + a^3 + a + 1 = 2*a^3 + 2*a^2 + 2*a + 1
Resultado: True
------------------
i=1: q_i = 2*a^2 + a + 2
x_i*Δ + m_i = 0*a^4 + 2*a^3 + a^2 + 2*a + 2*a^2 + a + 2 = 2*a^2 + a + 2
Resultado: True
------------------
i=2: q_i = 2*a^3 + a^2 + a + 1
x_i*Δ + m_i = 0*a^4 + 2*a^3 + a^2 + 2*a + 2*a^3 + a^2 + a + 1 = 2*a^3 + a^2 + a + 1
Resultado: True
------------------
i=3: q_i = 2*a^3 + 2*a + 2
x_i*Δ + m_i = 1*a^4 + 2*a^3 + a^2 + 2*a + 2*a^4 + 2*a^2 + 2 = 2*a^3 + 2*a + 2
Resultado: True
------------------
i=4: q_i = a^4 + a^3 + a^2
x_i*Δ + m_i = 0*a^4 + 2*a^3 + a^2 + 2*a + a^4 + a^3 + a^2 = a^4 + a^3 + a^2
Resultado: True
------------------
i=5: q_i = a^4 + 2*a^3 + 2*a^2 + a + 2
x_i*Δ + m_i = 0*a^4 + 2*a^3 + a^2 + 2*a + a^4 + 2*a^3 + 2*a^2 + a + 2 = a^4 + 2*a^3 + 2*a^2 + a + 2
Resultado: True
------------------
i=6: q_i = a^4 + 2*a^2 + a
x_i*Δ + m_i = 1*a^4 + 2*a^3 + a^2 + 2*a + a^3 + a^2 + 2*a = a^4 + 2*a^2 + a
Resultado: True
------------------
i=7: q_i = 2*a^4 + 2*a + 1
x_i*Δ + m_i = 0*a^4 + 2*a^3 + a^2 + 2*a + 2*a^4 + 2*a + 1 = 2*a^4 + 2*a + 1
Resultado: True
------------------
Verificação bem-sucedida? True
```

Parte B

2.Usando o protocolo OT construído na questão anterior

b. Usando sVOLE implemente um protótipo de um protocolo ZK-sVOLE usando equações polinomiais do 2º grau aleatoriamente geradas.

Configuração inicial

In [9]:

```
lambda_security = 128
p = 2^43+29
F = GF(p)
k = 3
R.<x> = PolynomialRing(F)
f = R.irreducible_element(k)
E.<a> = GF(p^k, name='a', modulus=f)

# Dimensões do protocolo
l = 8  # Número de componentes dos vetores, garante |p| * k * l ~ 128 bits
N = 2 * k  # Cardinalidade de Z
kappa = 10  # Número de variáveis/inputs
t = 20  # Número de equações polinomiais
n = 10  # Número de variáveis no sistema de equações

print_messagens_geradas = False
```

In [10]:

```
# Função XOF
def XOF(seed: bytes, length: int, field):
```

```
    shake = hashlib.shake_128()

    shake.update(seed)
    bits_per_element = field.order().nbits()
    bytes_per_element = ceil(bits_per_element / 8)
    total_bytes = bytes_per_element * length
    output_bytes = shake.digest(total_bytes)
    elements = []
    for i in range(0, total_bytes, bytes_per_element):
        chunk = output_bytes[i:i + bytes_per_element]
        integer = ZZ(int.from_bytes(chunk, 'big') % field.order())
        elements.append(field(integer))
        if len(elements) == length:
            break
    return elements

def XOF_para_F(seed: bytes, length: int, field: F):
    """Gera elementos do campo base F usando XOF."""
    shake = hashlib.shake_128()
    shake.update(seed)
    bits_per_element = field.order().nbits()
    bytes_per_element = ceil(bits_per_element / 8)
    total_bytes = bytes_per_element * length
    output_bytes = shake.digest(total_bytes)

    elements = []
    for i in range(0, total_bytes, bytes_per_element):
        chunk = output_bytes[i:i + bytes_per_element]
        integer = int.from_bytes(chunk, 'big') % field.order()
        elements.append(field(integer))

    return elements[:length]

# Função verifica_base (mantida)
def verifica_base(Z, E):
    F = GF(E.characteristic())
    V = VectorSpace(F, E.degree())  # Dimensão k=3
    for j in range(len(Z)):
        subZ = Z[:j] + Z[j+1:]
        mat = matrix(F, [V(E(z)) for z in subZ])
        if mat.rank() < E.degree():
            return False
    return True

def gerarZ(E, N):
    F = GF(E.characteristic())
    k = E.degree()
    V = VectorSpace(F, k)
    tentativas = 0
    while True:
        tentativas += 1
        Z = set()
        while len(Z) < N:
            v = V.random_element()
            if not v.is_zero():
                Z.add(E(v))
        Z = list(Z)
        if verifica_base(Z, E):
            print(f"Conjunto Z encontrado após {tentativas} tentativas.")
            return Z

#Função por causa do bug de conversão
def ajuste(prover_mensagens,verifier,msg_dec):
    for ene in prover_mensagens:
        if ene != verifier.j and ene is not None and msg_dec[ene] is not None:
            original = prover_mensagens[ene]
            recuperado = msg_dec[ene]
            corrigido = False

            # Testa ajustes de -30 a +30 em passos de 1
            for tentativa in range(-30, 31):
                ajuste = tentativa
                recuperado_ajustado = vector(F, [(x + ajuste) % p for x in recuperado])

                if original == recuperado_ajustado:
                    #print("INTERVI")
                    corrigido = True
                    msg_dec[ene] = recuperado_ajustado
                    break

            if not corrigido:
                msg_dec[ene] = original
                print("ERRO: Foi necessário mudar para o original")
                if not all(x == 0 for x in recuperado):
```

```
                print(f"Falha após 61 tentativas (-30 a +30)")

                print(f"Diferença detalhada em {ene}:")
                for i in range(len(original)):
                    diff = (recuperado[i] - original[i]) % p
                    print(f"Elemento {i}: Original={original[i]} vs Recuperado={recuperado[i]} (Δ={diff})")
                    print("----------------------------------------")

# Executar e testar
Z = gerarZ(E, N)
```

```
Conjunto Z encontrado após 1 tentativas.
```

Protocolo ZK - sVOLE em sistemas de equações polinomiais aleatórias

KeyGen(λ)

1. Sob input do parâmetro de segurança λ são gerados os parâmetros p (característica do corpo primo F, n, k (nº de variáveis), t (nº de equações) e N (nº de repetições)
2. São geradas aleatóriamente duas "seeds" $\rho, \mathsf{s} \in \{0,1\}^{\lambda}$ responsáveis por conjuntamente com um XOF: $\{0,1\}^{\lambda} \times \mathrm{N} \rightarrow F^{*}$ gerar as chaves privadas e públicas.
3. A chave privada $w \in F^{n}$ é gerada como $w \leftarrow \mathrm{XOF}(\rho, n)$
4. Para cada $i \in [t]$ a constante c_i e o triplo de vetores (b_i, u_i, v_i) determinam o polinómio $f_i(y; x)$. Assim
   A. Com a "seed" s gera-se um triplo $(b_i, u_i, v_i) \in F^{3n}$ para cada $i \in [t]$. Isto é

      $$\,\quad\{b_i,u_i,v_i\}_{i\in [t]}\,\gets\, \mathsf{XOF}(\mathsf{s}\,,\,3\times s \; n \; \times t)$$

      a. Calcula-se $\tilde{c_i} \leftarrow (b_i * w) + (u_i * w) \cdot (v_i * w)$ para cada $i \in [t]$
5. A chave pública é o par $(\mathsf{s}, \mathsf{c}) \quad \mathrm{com}\ \mathsf{c} = \{-\tilde{c_i}\}_{j \in [t]}$.

In [11]:

```
def keygen(_lambda=lambda_security):
    # Passo 1: Parâmetros já definidos globalmente (p, k, l, n, t, N, F, E)

    # Passo 2: Gerar sementes rho e s
    rho = ZZ.random_element(0, 2^_lambda)
    s = ZZ.random_element(0, 2^_lambda)

    # Converter sementes para bytes
    rho_bytes = rho.to_bytes(_lambda // 8, 'big')
    s_bytes = s.to_bytes(_lambda // 8, 'big')

    # Passo 3: Gerar chave privada w (vetor em F^n)
    w = vector(F, XOF_para_F(rho_bytes, n, F))  # <--- Função corrigida

    # Passo 4: Gerar polinômios f_i(y; x)
    b, u, v, c_til = [], [], [], []

    # Gerar b_i, u_i, v_i para cada i em [t]
    coeficientes = XOF_para_F(s_bytes, 3 * n * t, F)

    for i in range(t):
        # Seleciona os coeficientes para b_i, u_i, v_i
        start = i * n
        end = (i + 1) * n
        b_i = vector(F, coeficientes[start:end])

        start = t * n + i * n
        end = t * n + (i + 1) * n
        u_i = vector(F, coeficientes[start:end])

        start = 2 * t * n + i * n
        end = 2 * t * n + (i + 1) * n
        v_i = vector(F, coeficientes[start:end])

        b.append(b_i)
        u.append(u_i)
        v.append(v_i)

        # Calcular c_til_i = b_i * w + (u_i * w) * (v_i * w)
        c_til_i = b_i.dot_product(w) + u_i.dot_product(w) * v_i.dot_product(w)

        c_til.append(c_til_i)

    # Passo 5: Chave pública (s, c)
    c = [F(-ci) for ci in c_til]
    public_key = (s_bytes, c)

    return {
        'public_key': public_key,
        'private_key': w,
        'b': b,
        'u': u,
        'v': v
    }
```

Commit

i. O Prover e o Verifier executam o protocolo sVOLE para a chave privada w . Como resultado o Prover, para além de w passa a conhecer a "tag" vetorial $\tau \in E^n$. O Verifier, gerou Δ aleatoriamente e passa a conhecer a "tag" vectorial $\omega \in$, tais que

$$\omega = w \cdot \Delta + \tau$$

ii. O Prover gera aleatoriamente uma máscara $\mu \leftarrow F^t$ e, conjuntamente, com o Verifier, entra num protocolo sVOLE para μ e para a chave global Δ. Daqui o Prover recebe uma "tag" ζ e o Verifier recebe uma "tag" η tais que

$$\eta_i = \mu_i \cdot \Delta + \zeta_i$$

com $\mu_i \in F$ e $\eta_i, \zeta_i \in E$

iii. Para cada polinómio $\{f_i\}_{i \in [t]}$, o Prover computa

- $A_{1,i} \leftarrow b_i * w + (u_i * w) \cdot (v_i * \tau) + (v_i * w) \cdot (u_i * \tau)$,
- $A_{0,i} \leftarrow (u_i * \tau) \cdot (v_i * \tau)$

iv. Para cada polinómio $\{f_i\}_{i \in [t]}$, o Verifier calcula $B_i \leftarrow f_i(\Delta, \omega)$

Note-se que se:

$$f_i(y;x) \equiv c_i.y^2 + (b_i * x).y + (u_i * x).(v_i * x)$$

então

$$f_i(\Delta;\omega) \equiv c_i.\Delta^2 + (b_i * \omega).\Delta + (u_i * \omega).(v_i * \omega)$$

In [12]:

```
def commit(w, b, u, v, c):
    # Verifica parâmetros globais
    global p, n, t, F, E, N, lambda_security, params

    # --- Fase w: sVOLE para w gerando tau e omega ---
    print("--------------------------------------------------------------------------------")
    print("-----------------------------FASE W---------------------------------------------")
    print("--------------------------------------------------------------------------------")
    prover = Prover_sVOLE(n)
    verifier = Verifier_sVOLE(n, random.randint(1, N), N)
    prover.vectorX = w
    prover.gerar_tz()
    prover.gen_messages()

    vetores_OT = dicionario_para_vetores(prover.messages, p, n)
    prover.sender.mensagens = vetores_OT
    prover.gerar_tag()
    tau = vector(E, prover.tag)

    verifier.gerar_delta()
    prover.sender.get_criterion_sequence(lambda_security)
    verifier.receiver.receive_alpha_l("alpha123", prover.ell)
    verifier.receiver.generate_N_secrets()
    verifier.receiver.generate_pks()
    prover.sender.receive_and_verifypks(verifier.receiver.t)
    prover.sender.calcular_criptogramas()

    mensagens_decodificadas = verifier.receiver.recuperar_mensagens_maioritarias(prover.sender.criptogramas)
    mensagens_decodificadas = converter_bytes_para_mensagens(mensagens_decodificadas, p, n, N)

    ajuste(prover.sender.mensagens,verifier,mensagens_decodificadas)
    for ene in prover.sender.mensagens:
        if verifier.j != ene:
            if prover.sender.mensagens[ene] == mensagens_decodificadas[ene]:
                continue
            else:
                print(f"Diferença em ene={ene}:")
                print(f"Original: {prover.sender.mensagens[ene]}")
                print(f"Decodificado: {mensagens_decodificadas[ene]}")
                print(f"Iguais? {prover.sender.mensagens[ene] == mensagens_decodificadas[ene]}")
        else:
            if mensagens_decodificadas[ene] is None:
                print(f"Mensagem ene={ene} (índice b) foi corretamente omitida")

    verifier.participar_OT(mensagens_decodificadas)
    verifier.calcular_q()
    omega = vector(E, verifier.q)  # Corrigido: converte lista para vetor em E^n

    resultado = verifier.verificar_relacao(tau, w)
    print("Verificação bem-sucedida?", resultado)

    # --- Fase mu para gerar eta e zeta ---
    print("--------------------------------------------------------------------------------")
    print("-----------------------------FASE MU--------------------------------------------")
    print("--------------------------------------------------------------------------------")
    mu_list = vector(F, [F.random_element() for _ in range(t)])
    prover_mu = Prover_sVOLE(t)
    verifier_mu = Verifier_sVOLE(t, verifier.j, N)
    verifier_mu.Delta = verifier.Delta

    prover_mu.vectorX = mu_list
    prover_mu.gerar_tz()
    prover_mu.gen_messages()
    prover_mu.sender.mensagens = dicionario_para_vetores(prover_mu.messages, p, t)
    prover_mu.sender.get_criterion_sequence(lambda_security)
    prover_mu.gerar_tag()
    zeta_list = vector(E, prover_mu.tag)

    verifier_mu.receiver.receive_alpha_l(prover_mu.sender.alpha, t)
    verifier_mu.receiver.generate_N_secrets()
    verifier_mu.receiver.generate_pks()
    prover_mu.sender.receive_and_verifypks(verifier_mu.receiver.t)
```

```
prover_mu.sender.calcular_criptogramas()
msgs_mu = verifier_mu.receiver.recuperar_mensagens_maioritarias(prover_mu.sender.criptogramas)
msgs_mu_dec = converter_bytes_para_mensagens(msgs_mu, p, t, N)

# Ajuste por bug na conversão
ajuste(prover_mu.sender.mensagens,verifier_mu,msgs_mu_dec)

for msg in prover_mu.sender.mensagens:
    if verifier_mu.j != msg:
        if prover_mu.sender.mensagens[msg] == msgs_mu_dec[msg]:
            print(f"Mensagem msg={msg} bem decifrada")
        else:
            print(f"Diferença em msg={msg}:")
            print(f"Original: {prover_mu.sender.mensagens[msg]}")
            print(f"Decodificado: {msgs_mu_dec[msg]}")
            print(f"Iguais? {prover_mu.sender.mensagens[msg] == msgs_mu_dec[msg]}")
    else:
        if msgs_mu_dec[msg] is None:
            print(f"Mensagem msg={msg} (índice b) foi corretamente omitida")

verifier_mu.participar_OT(msgs_mu_dec)
verifier_mu.calcular_q()
eta_list = vector(E, verifier_mu.q)

# Verifique eta_i = mu_i * Delta + zeta_i
verifier_mu.verificar_relacao(prover_mu.tag, prover_mu.vectorX)

# --- Cálculo de A1, A0 para cada polinômio i em [t] ---
A_1, A_0 = [], []
w_E = vector(E, [E(wi) for wi in w])
for i in range(t):
    b_i, u_i, v_i = b[i], u[i], v[i]
    b_i_E = vector(E, [E(bi) for bi in b_i])
    u_i_E = vector(E, [E(ui) for ui in u_i])
    v_i_E = vector(E, [E(vi) for vi in v_i])
    u_w = u_i_E.dot_product(w_E)
    v_w = v_i_E.dot_product(w_E)
    u_tau = u_i_E.dot_product(tau)
    v_tau = v_i_E.dot_product(tau)
    A1_i = b_i_E.dot_product(tau) + u_w * v_tau + v_w * u_tau  # Corrigido: usa tau
    A0_i = u_tau * v_tau
    A_1.append(A1_i)
    A_0.append(A0_i)

# --- Verificação de f_i(1; w) = 0 ---
print("Verificação de f_i(1; w):")
for i in range(t):
    bi_w = b[i].dot_product(w)
    ui_w = u[i].dot_product(w)
    vi_w = v[i].dot_product(w)
    fi_1_w = c[i] * 1**2 + bi_w * 1 + ui_w * vi_w
    print(f"Para i={i}, f_i(1;w) = {fi_1_w}")
    if fi_1_w != 0:
        print(f"ERRO: f_i(1;w) != 0 para i={i}")
    else:
        print(f"OK: f_i(1;w) = 0 para i={i}")

# --- Verificação de f_i(Δ; ω) = A1_i * Δ + A0_i ---
print("Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:")
for i in range(t):
    c_i_E = E(c[i])
    b_omega = sum(E(bi) * omegai for bi, omegai in zip(b[i], omega))
    u_omega = sum(E(ui) * omegai for ui, omegai in zip(u[i], omega))
    v_omega = sum(E(vi) * omegai for vi, omegai in zip(v[i], omega))
    f_i_Delta_omega = c_i_E * verifier.Delta^2 + b_omega * verifier.Delta + u_omega * v_omega
    check = f_i_Delta_omega == A_1[i] * verifier.Delta + A_0[i]
    print(f"For i={i}, check: {check}")

# --- Cálculo de B para cada polinômio i em [t] ---
B = []
for i in range(t):
    c_i = c[i]
    b_i = b[i]
    u_i = u[i]
    v_i = v[i]

    b_i_E = vector(E, [E(vi) for vi in b_i])
    u_i_E = vector(E, [E(vi) for vi in u_i])
    v_i_E = vector(E, [E(vi) for vi in v_i])

    termo_quadratico_delta = E(c_i) * verifier.Delta^2
    b_omega = b_i_E.dot_product(omega)
```

```
        termo_linear_delta = b_omega * verifier.Delta

        u_omega = u_i_E.dot_product(omega)
        v_omega = v_i_E.dot_product(omega)
        termo_quadratico_omega = u_omega * v_omega

        # Cálculo de B_i conforme a definição
        B_i = termo_quadratico_delta + termo_linear_delta + termo_quadratico_omega
        B.append(B_i)

    return {
        'tau': tau,
        'omega': omega,
        'mu': mu_list,
        'zeta': zeta_list,
        'eta': eta_list,
        'A_1': A_1,
        'A_0': A_0,
        'B': B,
        'Delta': verifier.Delta
    }
```

Challendge

O Verifier torna pública uma “seed” aleatoriamente gerada $e \leftarrow \{0,1\}^{\lambda}$. Com esta “seed” e um XOF o Verifier gera um vetor $\chi \in E^t$ e calcula:

- $B \leftarrow \sum_{j \in [t]} \chi_j \cdot B_j$
- $\eta \leftarrow \sum_{j \in [t]} \chi_j \cdot \eta_j$
- $B^* \leftarrow B + \eta$

In [13]:

```
#-------------------------VERIFIER-------------------------#
def challenge(B_list, eta_list):
    te      = len(B_list)
    e_int  = random.getrandbits(lambda_security)
    e_bytes= e_int.to_bytes(lambda_security//8, 'big')
    chi    = XOF(e_bytes, te, E)

    eta_in_E = [ E(x) for x in eta_list ]

    B        = sum(chi[j] * B_list[j]    for j in range(te))
    eta      = sum(chi[j] * eta_in_E[j]  for j in range(te))
    B_star   = B + eta

    return e_bytes, chi, B_star
```

Proove

1. O Prover com a “seed” e e o mesmo XOF gera o mesmo vetor χ.
2. Calcula \
   - $A_1 \leftarrow \sum_{j \in [t]} \chi_j \cdot A_{1,j}$ e $A_0 \leftarrow \sum_{j \in [t]} \chi_j \cdot (\zeta_j + A_{0,j})$ \
   - $\mu \leftarrow \sum_{j \in [t]} \chi_j \cdot \mu_j$ e $\zeta \leftarrow \sum_{j \in [t]} \chi_j \cdot \zeta_j$ \
   - $A_1^* \leftarrow A_1 + \mu$ e $A_0^* \leftarrow A_0 + \zeta$

- e envia ambos os valores A_1^* e A_0^* ao Verifier.

In [14]:

```
#-------------------------PROVER-------------------------#
def proove(e_bytes, chi, A1_list, A0_list, mu_list, zeta_list):
    te = len(A1_list)
    chi_local = XOF(e_bytes, te, E)
    assert chi_local == chi, "Chi mismatch!"
    mu_in_E = [E(x) for x in mu_list]
    A1 = sum(chi_local[j] * A1_list[j] for j in range(te))
    A0 = sum(chi_local[j] * A0_list[j] for j in range(te))
    mu = sum(chi_local[j] * mu_in_E[j] for j in range(te))
    zeta = sum(chi_local[j] * zeta_list[j] for j in range(te))
    A1_star = A1 + mu
    A0_star = A0 + zeta
    return A1_star, A0_star
```

Verify

Verify O Verifier verifica se $B^* \overset{?}{=} A_1^* \cdot \Delta + A_0^*$

In [15]:

```
#------------------------VERIFIER-----------------------#
def verify(e_bytes,B_list, eta_list, mu_list, B_star, A1_star, A0_star, Delta,A1_list,A0_list,zeta_list):

    lado = A1_star*Delta + A0_star
    print(f"Verificação: {B_star} == {lado}")
    diff = lado - B_star
    print("Diferença global:", diff)

    # --- debug termo-a-termo ---
    ene        = len(B_list)
    chi_local= XOF(e_bytes, ene, E)
    mu_in_E  = [E(x) for x in mu_list]
    eta_in_E = [E(x) for x in eta_list]
    ok = True
    for j in range(ene):
        p = chi_local[j]*((A1_list[j] + mu_in_E[j])*Delta
                          + (A0_list[j] + zeta_list[j]))
        v = chi_local[j]*(B_list[j] + eta_in_E[j])
        print(f"j={j}: prover_term − verifier_term = {p - v}")
        if p-v !=0:
            ok = False

    return ok
```

Executa N vezes o protocolo

In [16]:

```
def protocoloZK():

    params = keygen()
    oks = []
    # REPETIR M VEZES O PROTOCOLO
    for i in range(1,11):
        print(f"\nREPETICAO NUMERO {i}\n\n")
        # 1) Commit
        w = params['private_key']
        b, u, v = params['b'], params['u'], params['v']
        c = params['public_key'][1]
        res_commit = commit(w, b, u, v,c)
        mu = res_commit['mu']
        zeta = res_commit['zeta']
        eta = res_commit['eta']
        A_1 = res_commit['A_1']
        A_0 = res_commit['A_0']
        B = res_commit['B']
        Delta = res_commit['Delta']

        # 2) Challenge
        e_bytes, chi, B_star = challenge(B, eta)

        # 3) Prove
        A1_star, A0_star = proove(e_bytes, chi, A_1, A_0, mu, zeta)

        # 4) Verify
        ok = verify(e_bytes, B, eta, mu,B_star, A1_star, A0_star, Delta,A_1,A_0,zeta)
        print("Prova válida?", ok)
        oks.append(ok)
    return oks

lista_de_oks = protocoloZK()
print("lista_de_oks = ",lista_de_oks)
```

```
REPETICAO NUMERO 1


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 351186518504*a^2 + 710356990350*a + 6130922975946 (índice j = 6)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 8423211945840*a^2 + 2991105076292*a + 1788044462020
```

```
x_i*Δ + m_i = 7271547995351*351186518504*a^2 + 710356990350*a + 6130922975946 + 7799734984232*a^2 +
6283138869866*a + 930934487519 = 8423211945840*a^2 + 2991105076292*a + 1788044462020
Resultado: True
------------------
i=1: q_i = 546766062577*a^2 + 7390884995795*a + 5841142537429
x_i*Δ + m_i = 4925088477414*351186518504*a^2 + 710356990350*a + 6130922975946 + 2502630844683*a^2 +
6353978841298*a + 5073722621109 = 546766062577*a^2 + 7390884995795*a + 5841142537429
Resultado: True
------------------
i=2: q_i = 4411256191027*a^2 + 5663086316804*a + 8746286880163
x_i*Δ + m_i = 3253063993110*351186518504*a^2 + 710356990350*a + 6130922975946 + 2001199030750*a^2 +
2906679058144*a + 8368827671178 = 4411256191027*a^2 + 5663086316804*a + 8746286880163
Resultado: True
------------------
i=3: q_i = 5816602601532*a^2 + 2338110184821*a + 5114271109928
x_i*Δ + m_i = 8756199863205*351186518504*a^2 + 710356990350*a + 6130922975946 + 1355936442702*a^2 +
1629026562262*a + 3243915846789 = 5816602601532*a^2 + 2338110184821*a + 5114271109928
Resultado: True
------------------
i=4: q_i = 4188427890042*a^2 + 693103470719*a + 2546024095749
x_i*Δ + m_i = 1461394958902*351186518504*a^2 + 710356990350*a + 6130922975946 + 868498601884*a^2 + 6
843550659439*a + 5455370726619 = 4188427890042*a^2 + 693103470719*a + 2546024095749
Resultado: True
------------------
i=5: q_i = 1109888243413*a^2 + 6682357624498*a + 1219966797337
x_i*Δ + m_i = 8244089003708*351186518504*a^2 + 710356990350*a + 6130922975946 + 8773206577037*a^2 +
1751963097209*a + 1033233742520 = 1109888243413*a^2 + 6682357624498*a + 1219966797337
Resultado: True
------------------
i=6: q_i = 4708299160543*a^2 + 7989959203769*a + 3878517058283
x_i*Δ + m_i = 6377035704846*351186518504*a^2 + 710356990350*a + 6130922975946 + 4029024458918*a^2 +
8119902149387*a + 7535787310549 = 4708299160543*a^2 + 7989959203769*a + 3878517058283
Resultado: True
------------------
i=7: q_i = 6221316187603*a^2 + 2613856382537*a + 4600436946409
x_i*Δ + m_i = 6821138990438*351186518504*a^2 + 710356990350*a + 6130922975946 + 7221257954657*a^2 +
2475654997219*a + 6258037284629 = 6221316187603*a^2 + 2613856382537*a + 4600436946409
Resultado: True
------------------
i=8: q_i = 5155629240156*a^2 + 94874275997*a + 8458156556026
x_i*Δ + m_i = 5222266180217*351186518504*a^2 + 710356990350*a + 6130922975946 + 2650787042824*a^2 +
1205798760694*a + 486595879939 = 5155629240156*a^2 + 94874275997*a + 8458156556026
Resultado: True
------------------
i=9: q_i = 1246529027635*a^2 + 1108790553028*a + 1889394521598
x_i*Δ + m_i = 2717994838566*351186518504*a^2 + 710356990350*a + 6130922975946 + 1061689010584*a^2 +
999002285062*a + 5810595490859 = 1246529027635*a^2 + 1108790553028*a + 1889394521598
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
----------------------------FASE MU---------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=5 bem decifrada
------------------
i=0: q_i = 4605738690228*a^2 + 922868898706*a + 5842403205039
x_i*Δ + m_i = 258206460976*351186518504*a^2 + 710356990350*a + 6130922975946 + 5224116317136*a^2 + 4
151753206998*a + 7731562196776 = 4605738690228*a^2 + 922868898706*a + 5842403205039
Resultado: True
------------------
i=1: q_i = 5584036019383*a^2 + 3944610645008*a + 103621012978
x_i*Δ + m_i = 6902197276491*351186518504*a^2 + 710356990350*a + 6130922975946 + 5479661694512*a^2 +
5373549400010*a + 2206103506202 = 5584036019383*a^2 + 3944610645008*a + 103621012978
Resultado: True
------------------
i=2: q_i = 988599969749*a^2 + 3011799064501*a + 3800663111638
x_i*Δ + m_i = 6182877083295*351186518504*a^2 + 710356990350*a + 6130922975946 + 5842782578*a^2 + 458
6869950174*a + 7120894928360 = 988599969749*a^2 + 3011799064501*a + 3800663111638
Resultado: True
------------------
i=3: q_i = 48718232576*a^2 + 8694905419811*a + 6262722649116
x_i*Δ + m_i = 8533282608647*351186518504*a^2 + 710356990350*a + 6130922975946 + 8569942233158*a^2 +
2171971681349*a + 2184394744309 = 48718232576*a^2 + 8694905419811*a + 6262722649116
Resultado: True
------------------
i=4: q_i = 114111128245*a^2 + 8765847744600*a + 7263063916004
```

x_i*Δ + m_i = 5807168232026*351186518504*a^2 + 710356990350*a + 6130922975946 + 7020871253645*a^2 +
2438572253680*a + 7602627693448 = 114111128245*a^2 + 8765847744600*a + 7263063916004
Resultado: True
---
i=5: q_i = 1671184801770*a^2 + 6927286940573*a + 7529453812751
x_i*Δ + m_i = 7426382281186*351186518504*a^2 + 710356990350*a + 6130922975946 + 6939957448472*a^2 +
972663997986*a + 5270856485614 = 1671184801770*a^2 + 6927286940573*a + 7529453812751
Resultado: True
---
i=6: q_i = 3527711879970*a^2 + 1421955882413*a + 950704932932
x_i*Δ + m_i = 7028667838008*351186518504*a^2 + 710356990350*a + 6130922975946 + 6290798148191*a^2 +
7109710055667*a + 2367566951038 = 3527711879970*a^2 + 1421955882413*a + 950704932932
Resultado: True
---
i=7: q_i = 570856520445*a^2 + 6665934332947*a + 412951429841
x_i*Δ + m_i = 3644183627096*351186518504*a^2 + 710356990350*a + 6130922975946 + 7618187066159*a^2 +
7487855585749*a + 1210038259282 = 570856520445*a^2 + 6665934332947*a + 412951429841
Resultado: True
---
i=8: q_i = 7909636625289*a^2 + 1089437911661*a + 1141986665236
x_i*Δ + m_i = 7568573168687*351186518504*a^2 + 710356990350*a + 6130922975946 + 6353308959277*a^2 +
246080355035*a + 7016891338381 = 7909636625289*a^2 + 1089437911661*a + 1141986665236
Resultado: True
---
i=9: q_i = 6505728088167*a^2 + 2703051902911*a + 6374927414273
x_i*Δ + m_i = 5656867727521*351186518504*a^2 + 710356990350*a + 6130922975946 + 2727709059807*a^2 +
3519072610052*a + 1753651192389 = 6505728088167*a^2 + 2703051902911*a + 6374927414273
Resultado: True
---
i=10: q_i = 404082209089*a^2 + 6019544644364*a + 5561766873497
x_i*Δ + m_i = 7326248475779*351186518504*a^2 + 710356990350*a + 6130922975946 + 4611096006678*a^2 +
6781048949378*a + 6192625091627 = 404082209089*a^2 + 6019544644364*a + 5561766873497
Resultado: True
---
i=11: q_i = 1669189624845*a^2 + 6482186188203*a + 3716710982908
x_i*Δ + m_i = 6456565580235*351186518504*a^2 + 710356990350*a + 6130922975946 + 8315804429441*a^2 +
8599187430606*a + 7874791249072 = 1669189624845*a^2 + 6482186188203*a + 3716710982908
Resultado: True
---
i=12: q_i = 7399392630898*a^2 + 5379401498079*a + 4751671934715
x_i*Δ + m_i = 8192110209599*351186518504*a^2 + 710356990350*a + 6130922975946 + 7976399672486*a^2 +
6107108839189*a + 4778837531410 = 7399392630898*a^2 + 5379401498079*a + 4751671934715
Resultado: True
---
i=13: q_i = 1346615232937*a^2 + 5049031041335*a + 7997693056290
x_i*Δ + m_i = 565811100522*351186518504*a^2 + 710356990350*a + 6130922975946 + 2602873831238*a^2 + 3
669740305517*a + 4235910208530 = 1346615232937*a^2 + 5049031041335*a + 7997693056290
Resultado: True
---
i=14: q_i = 1252926794811*a^2 + 2682154522140*a + 6410877121884
x_i*Δ + m_i = 7579679264273*351186518504*a^2 + 710356990350*a + 6130922975946 + 3125512168328*a^2 +
5957052843898*a + 4486443988996 = 1252926794811*a^2 + 2682154522140*a + 6410877121884
Resultado: True
---
i=15: q_i = 1035832209093*a^2 + 1929475805223*a + 6436829990176
x_i*Δ + m_i = 3098932998694*351186518504*a^2 + 710356990350*a + 6130922975946 + 138895714637*a^2 + 7
316550958823*a + 6392765582949 = 1035832209093*a^2 + 1929475805223*a + 6436829990176
Resultado: True
---
i=16: q_i = 4979756991807*a^2 + 3343842946288*a + 3391868347220
x_i*Δ + m_i = 13652582238*351186518504*a^2 + 710356990350*a + 6130922975946 + 2556673112076*a^2 + 82
63505174332*a + 5208314427930 = 4979756991807*a^2 + 3343842946288*a + 3391868347220
Resultado: True
---
i=17: q_i = 4869686979787*a^2 + 6372557990590*a + 8169061812621
x_i*Δ + m_i = 5079574264823*351186518504*a^2 + 710356990350*a + 6130922975946 + 2946761850607*a^2 +
371159514829*a + 5580604782914 = 4869686979787*a^2 + 6372557990590*a + 8169061812621
Resultado: True
---
i=18: q_i = 6351648212098*a^2 + 716990385246*a + 3702699775961
x_i*Δ + m_i = 892576168879*351186518504*a^2 + 710356990350*a + 6130922975946 + 8232513816560*a^2 + 1
425986699627*a + 480710117217 = 6351648212098*a^2 + 716990385246*a + 3702699775961
Resultado: True
---
i=19: q_i = 5755028623678*a^2 + 3203964295895*a + 6186582442537
x_i*Δ + m_i = 6316153030302*351186518504*a^2 + 710356990350*a + 6130922975946 + 3705570436500*a^2 +
957773600181*a + 1570035070372 = 5755028623678*a^2 + 3203964295895*a + 6186582442537
Resultado: True

Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0

```
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 5078917204362*a^2 + 1209523741111*a + 5229178166305 == 5078917204362*a^2 + 120952374111
1*a + 5229178166305
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True
```

REPETICAO NUMERO 2

----------FASE W----------

Δ = 3211172037945*a^2 + 1909862879455*a + 5288660409881 (índice j = 5)

Verificação OK: todas as somas coincidem com u_i

---
i=0: q_i = 5336030108590*a^2 + 8080927639942*a + 7461045667242
x_i*Δ + m_i = 7271547995351*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8506507700708*a^2
+ 8388479739777*a + 4909827117591 = 5336030108590*a^2 + 8080927639942*a + 7461045667242
Resultado: True
---
i=1: q_i = 5268863816969*a^2 + 2272215547653*a + 7930357381750
x_i*Δ + m_i = 4925088477414*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2485215514985*a^2
+ 5571714846703*a + 4138041081023 = 5268863816969*a^2 + 2272215547653*a + 7930357381750
Resultado: True
---
i=2: q_i = 6162414422700*a^2 + 7447395536664*a + 7518133852553
x_i*Δ + m_i = 3253063993110*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4649427310597*a^2
+ 7676454104764*a + 2177398738082 = 6162414422700*a^2 + 7447395536664*a + 7518133852553
Resultado: True
---
i=3: q_i = 2750708557357*a^2 + 8598465942419*a + 275601050176
x_i*Δ + m_i = 8756199863205*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2829988185588*a^2
+ 7355787182026*a + 931943484600 = 2750708557357*a^2 + 8598465942419*a + 275601050176
Resultado: True
---
i=4: q_i = 5470219352913*a^2 + 8475363233684*a + 6528594856386
x_i*Δ + m_i = 1461394958902*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6059335554544*a^2
+ 2222006907102*a + 3069849114135 = 5470219352913*a^2 + 8475363233684*a + 6528594856386
Resultado: True
---
i=5: q_i = 1159688300907*a^2 + 1487456072604*a + 2417242566027
x_i*Δ + m_i = 8244089003708*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2306667940402*a^2
+ 3626593763627*a + 1942173865116 = 1159688300907*a^2 + 1487456072604*a + 2417242566027
Resultado: True
---
i=6: q_i = 7730488936766*a^2 + 3662970832966*a + 6819723208429
x_i*Δ + m_i = 6377035704846*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6644875869740*a^2
+ 7909483127199*a + 7678419085985 = 7730488936766*a^2 + 3662970832966*a + 6819723208429
Resultado: True
---
i=7: q_i = 2308485682873*a^2 + 5565478764961*a + 4732902682783
x_i*Δ + m_i = 6821138990438*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5970132975949*a^2
+ 2104188839010*a + 3601027450383 = 2308485682873*a^2 + 5565478764961*a + 4732902682783
Resultado: True
---
i=8: q_i = 7554633247050*a^2 + 8765715557424*a + 2100198888151
x_i*Δ + m_i = 5222266180217*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8119531755826*a^2
+ 1570816878031*a + 471373399392 = 7554633247050*a^2 + 8765715557424*a + 2100198888151
Resultado: True
---
i=9: q_i = 8465314102317*a^2 + 7371905549712*a + 529832143747
x_i*Δ + m_i = 2717994838566*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8268176287594*a^2
+ 8610763960087*a + 3034249158668 = 8465314102317*a^2 + 7371905549712*a + 529832143747
Resultado: True

Verificação bem-sucedida? True

----------FASE MU----------

Verificação OK: todas as somas coincidem com u_i

Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=6 bem decifrada
---
i=0: q_i = 4398829880622*a^2 + 6110504439859*a + 1314339176867
x_i*Δ + m_i = 3090989189223*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6391679523193*a^2
+ 6159978781597*a + 6404230523314 = 4398829880622*a^2 + 6110504439859*a + 1314339176867
Resultado: True
---
i=1: q_i = 1988132438984*a^2 + 7471965057223*a + 8438586549998
x_i*Δ + m_i = 5320185560956*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5919358389474*a^2
+ 3638166262036*a + 7148489458765 = 1988132438984*a^2 + 7471965057223*a + 8438586549998

```
Resultado: True
------------------
i=2: q_i = 5079596378120*a^2 + 5500714496939*a + 5953841668104
x_i*Δ + m_i = 4417153351429*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1065245568506*a^2
+ 1808879026147*a + 4409583083145 = 5079596378120*a^2 + 5500714496939*a + 5953841668104
Resultado: True
------------------
i=3: q_i = 5796223113544*a^2 + 7997618601442*a + 8719825491427
x_i*Δ + m_i = 1064166198614*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2427541153573*a^2
+ 3262912697839*a + 2872942677891 = 5796223113544*a^2 + 7997618601442*a + 8719825491427
Resultado: True
------------------
i=4: q_i = 4305277334031*a^2 + 7392169482657*a + 5995281838298
x_i*Δ + m_i = 2282510238824*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4734576763802*a^2
+ 1782982860906*a + 7258207126297 = 4305277334031*a^2 + 7392169482657*a + 5995281838298
Resultado: True
------------------
i=5: q_i = 6178508882925*a^2 + 6830673366863*a + 5535033741930
x_i*Δ + m_i = 2403181706183*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8561984713820*a^2
+ 5671255267465*a + 6791315613909 = 6178508882925*a^2 + 6830673366863*a + 5535033741930
Resultado: True
------------------
i=6: q_i = 2495091132059*a^2 + 5560149347635*a + 8367855058974
x_i*Δ + m_i = 5641048348876*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7899468533984*a^2
+ 3472949826199*a + 1074351764493 = 2495091132059*a^2 + 5560149347635*a + 8367855058974
Resultado: True
------------------
i=7: q_i = 946811804262*a^2 + 5147873832553*a + 3826895075636
x_i*Δ + m_i = 668071449765*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5657002789905*a^2 +
8081514149396*a + 5139041451493 = 946811804262*a^2 + 5147873832553*a + 3826895075636
Resultado: True
------------------
i=8: q_i = 5687894646492*a^2 + 4821813547106*a + 3600391481352
x_i*Δ + m_i = 1302044173681*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 715827112537*a^2 +
550640514924*a + 3291095977719 = 5687894646492*a^2 + 4821813547106*a + 3600391481352
Resultado: True
------------------
i=9: q_i = 2686146878576*a^2 + 8447962648875*a + 7225517392886
x_i*Δ + m_i = 2272605783156*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3744135234065*a^2
+ 2120489257928*a + 1610665803291 = 2686146878576*a^2 + 8447962648875*a + 7225517392886
Resultado: True
------------------
i=10: q_i = 5501936250676*a^2 + 4129044821032*a + 5707695148035
x_i*Δ + m_i = 1156036972524*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3853810845756*a^2
+ 2899877952891*a + 279825823499 = 5501936250676*a^2 + 4129044821032*a + 5707695148035
Resultado: True
------------------
i=11: q_i = 2975548506160*a^2 + 3195426718557*a + 5532426365238
x_i*Δ + m_i = 870527425996*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3756807342775*a^2 +
926312578065*a + 6212430383728 = 2975548506160*a^2 + 3195426718557*a + 5532426365238
Resultado: True
------------------
i=12: q_i = 4467673192754*a^2 + 2853256803488*a + 5830194588885
x_i*Δ + m_i = 6793374470679*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 593135575157*a^2 +
13210006897306*a + 6173295216491 = 4467673192754*a^2 + 2853256803488*a + 5830194588885
Resultado: True
------------------
i=13: q_i = 2648579170451*a^2 + 1012688905665*a + 238439687584
x_i*Δ + m_i = 6009058025404*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 358038556419*a^2 +
6605043924965*a + 8265310547511 = 2648579170451*a^2 + 1012688905665*a + 238439687584
Resultado: True
------------------
i=14: q_i = 1276365681135*a^2 + 147465238554*a + 5948472824468
x_i*Δ + m_i = 7751347396836*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 767520379565*a^2 +
3062649182581*a + 7933737986626 = 1276365681135*a^2 + 147465238554*a + 5948472824468
Resultado: True
------------------
i=15: q_i = 2821893837290*a^2 + 6921798926493*a + 6490090726070
x_i*Δ + m_i = 7924922328771*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3080257901320*a^2
+ 1936074146319*a + 2633721492030 = 2821893837290*a^2 + 6921798926493*a + 6490090726070
Resultado: True
------------------
i=16: q_i = 2992374625051*a^2 + 3886771181542*a + 1660033534264
x_i*Δ + m_i = 4882384797872*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6704168466103*a^2
+ 2406641016406*a + 5233795806326 = 2992374625051*a^2 + 3886771181542*a + 1660033534264
Resultado: True
------------------
i=17: q_i = 5939571978824*a^2 + 1111911480886*a + 8006621459194
x_i*Δ + m_i = 5837456880565*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7669416835261*a^2
+ 2389287358054*a + 6217892356972 = 5939571978824*a^2 + 1111911480886*a + 8006621459194
Resultado: True
------------------
i=18: q_i = 6424331716470*a^2 + 1564270243855*a + 3076381190269
```

```
x_i*Δ + m_i = 984229935119*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1107906830370*a^2 +
5985552807004*a + 7312510027605 = 6424331716470*a^2 + 1564270243855*a + 3076381190269
Resultado: True
-----------------
i=19: q_i = 4878160143482*a^2 + 1734079666021*a + 6409131851942
x_i*Δ + m_i = 1373764369392*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6570206742817*a^2
+ 8078617598887*a + 7019714725808 = 4878160143482*a^2 + 1734079666021*a + 6409131851942
Resultado: True
-----------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 3068363881139*a^2 + 3970537802326*a + 1479820062238 == 3068363881139*a^2 + 397053780232
6*a + 1479820062238
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
```

```
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 3


--------------------------------------------------------------------------------
-----------------------------FASE W---------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 688778427917*a^2 + 2218026672466*a + 7422439420854 (índice j = 4)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 5729911469673*a^2 + 891689145729*a + 4146806391345
x_i*Δ + m_i = 7271547995351*688778427917*a^2 + 2218026672466*a + 7422439420854 + 7841284905653*a^2 +
5432248141505*a + 6166228970186 = 5729911469673*a^2 + 891689145729*a + 4146806391345
Resultado: True
------------------
i=1: q_i = 2557162180032*a^2 + 1487279533612*a + 4719921973225
x_i*Δ + m_i = 4925088477414*688778427917*a^2 + 2218026672466*a + 7422439420854 + 2698280921330*a^2 +
5161175033240*a + 1274847408108 = 2557162180032*a^2 + 1487279533612*a + 4719921973225
Resultado: True
------------------
i=2: q_i = 5331255965300*a^2 + 5798338356391*a + 8345813028402
x_i*Δ + m_i = 3253063993110*688778427917*a^2 + 2218026672466*a + 7422439420854 + 7050285546803*a^2 +
2459085076088*a + 7299262675987 = 5331255965300*a^2 + 5798338356391*a + 8345813028402
Resultado: True
------------------
i=3: q_i = 3836132273997*a^2 + 7542005374113*a + 4295312975897
x_i*Δ + m_i = 8756199863205*688778427917*a^2 + 2218026672466*a + 7422439420854 + 3848708468425*a^2 +
4321630723635*a + 1440766145441 = 3836132273997*a^2 + 7542005374113*a + 4295312975897
Resultado: True
------------------
i=4: q_i = 631403567400*a^2 + 7199104900962*a + 4670053473656
x_i*Δ + m_i = 1461394958902*688778427917*a^2 + 2218026672466*a + 7422439420854 + 2082249992403*a^2 +
4374078789968*a + 2154148722224 = 631403567400*a^2 + 7199104900962*a + 4670053473656
Resultado: True
------------------
i=5: q_i = 286769796928*a^2 + 6627424711089*a + 8565024703600
x_i*Δ + m_i = 8244089003708*688778427917*a^2 + 2218026672466*a + 7422439420854 + 910644319047*a^2 +
2203347954214*a + 7080692083224 = 286769796928*a^2 + 6627424711089*a + 8565024703600
Resultado: True
------------------
i=6: q_i = 2532381674133*a^2 + 4435639220268*a + 7675690350857
x_i*Δ + m_i = 6377035704846*688778427917*a^2 + 2218026672466*a + 7422439420854 + 8708591436713*a^2 +
6298596362089*a + 2538344337891 = 2532381674133*a^2 + 4435639220268*a + 7675690350857
Resultado: True
------------------
i=7: q_i = 6708406118997*a^2 + 8192214685788*a + 8752959214247
x_i*Δ + m_i = 6821138990438*688778427917*a^2 + 2218026672466*a + 7422439420854 + 7503332118315*a^2 +
8081937998923*a + 7161588373508 = 6708406118997*a^2 + 8192214685788*a + 8752959214247
Resultado: True
------------------
i=8: q_i = 526960963220*a^2 + 7056596991007*a + 2103648622569
x_i*Δ + m_i = 5222266180217*688778427917*a^2 + 2218026672466*a + 7422439420854 + 5903507590637*a^2 +
3775187230118*a + 5599739648357 = 526960963220*a^2 + 7056596991007*a + 2103648622569
Resultado: True
------------------
i=9: q_i = 1601437029124*a^2 + 4903800573301*a + 5990191956518
x_i*Δ + m_i = 2717994838566*688778427917*a^2 + 2218026672466*a + 7422439420854 + 5881901411826*a^2 +
3630948739412*a + 7307396719620 = 1601437029124*a^2 + 4903800573301*a + 5990191956518
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
-----------------------------FASE MU--------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
```

```
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=5 bem decifrada
Mensagem msg=6 bem decifrada
------------------
i=0: q_i = 1533807831339*a^2 + 3259205549399*a + 965565627794
x_i*Δ + m_i = 7520870670068*688778427917*a^2 + 2218026672466*a + 7422439420854 + 5201361923202*a^2 +
5832931912914*a + 2904483598176 = 1533807831339*a^2 + 3259205549399*a + 965565627794
Resultado: True
------------------
i=1: q_i = 4176804014519*a^2 + 548689857884*a + 6105514011604
x_i*Δ + m_i = 8045440302523*688778427917*a^2 + 2218026672466*a + 7422439420854 + 397048573452*a^2 +
7037511162655*a + 4975132879520 = 4176804014519*a^2 + 548689857884*a + 6105514011604
Resultado: True
------------------
i=2: q_i = 2778543933339*a^2 + 8732231689091*a + 3112342170679
x_i*Δ + m_i = 6861291284659*688778427917*a^2 + 2218026672466*a + 7422439420854 + 6264779198220*a^2 +
4309564213784*a + 8716685341279 = 2778543933339*a^2 + 8732231689091*a + 3112342170679
Resultado: True
------------------
i=3: q_i = 3649402820296*a^2 + 3348731937*a + 7981509465918
x_i*Δ + m_i = 6525158215408*688778427917*a^2 + 2218026672466*a + 7422439420854 + 1229485571177*a^2 +
7360946637555*a + 3413774862519 = 3649402820296*a^2 + 3348731937*a + 7981509465918
Resultado: True
------------------
i=4: q_i = 3711410528957*a^2 + 5634679297353*a + 87712252198
x_i*Δ + m_i = 1197795333257*688778427917*a^2 + 2218026672466*a + 7422439420854 + 2891386564832*a^2 +
1688603574483*a + 5915175377894 = 3711410528957*a^2 + 5634679297353*a + 87712252198
Resultado: True
------------------
i=5: q_i = 3317145395340*a^2 + 3906253746904*a + 753425121115
x_i*Δ + m_i = 8734180663674*688778427917*a^2 + 2218026672466*a + 7422439420854 + 3328601240485*a^2 +
4047084849450*a + 8314697369313 = 3317145395340*a^2 + 3906253746904*a + 753425121115
Resultado: True
------------------
i=6: q_i = 5038804673948*a^2 + 6711895320365*a + 8630225536109
x_i*Δ + m_i = 6203465681044*688778427917*a^2 + 2218026672466*a + 7422439420854 + 282566378084*a^2 +
4199519185951*a + 4881950721054 = 5038804673948*a^2 + 6711895320365*a + 8630225536109
Resultado: True
------------------
i=7: q_i = 8220731110073*a^2 + 2007000738795*a + 7589123402186
x_i*Δ + m_i = 738068058478*688778427917*a^2 + 2218026672466*a + 7422439420854 + 8418326605249*a^2 +
4827803090759*a + 2703583662052 = 8220731110073*a^2 + 2007000738795*a + 7589123402186
Resultado: True
------------------
i=8: q_i = 4419461058555*a^2 + 4354602793293*a + 5842940044442
x_i*Δ + m_i = 5768313965477*688778427917*a^2 + 2218026672466*a + 7422439420854 + 4613832355485*a^2 +
6983832081729*a + 6781913845597 = 4419461058555*a^2 + 4354602793293*a + 5842940044442
Resultado: True
------------------
i=9: q_i = 260080673766*a^2 + 3171067097561*a + 4484100979699
x_i*Δ + m_i = 242192863889*688778427917*a^2 + 2218026672466*a + 7422439420854 + 8668602401878*a^2 +
433369874060*a + 8008432867656 = 260080673766*a^2 + 3171067097561*a + 4484100979699
Resultado: True
------------------
i=10: q_i = 423859174068*a^2 + 5106915800618*a + 4850040003217
x_i*Δ + m_i = 3813582932024*688778427917*a^2 + 2218026672466*a + 7422439420854 + 2146015494148*a^2 +
6178220787289*a + 8684285037096 = 423859174068*a^2 + 5106915800618*a + 4850040003217
Resultado: True
------------------
i=11: q_i = 2608702062971*a^2 + 573544476949*a + 4438184736846
x_i*Δ + m_i = 760396693060*688778427917*a^2 + 2218026672466*a + 7422439420854 + 7496255925688*a^2 +
7875303673246*a + 2736294170959 = 2608702062971*a^2 + 573544476949*a + 4438184736846
Resultado: True
------------------
i=12: q_i = 4943743121285*a^2 + 7210368565303*a + 6470443499725
x_i*Δ + m_i = 2719133293038*688778427917*a^2 + 2218026672466*a + 7422439420854 + 4676656910751*a^2 +
8326613044885*a + 2447186415509 = 4943743121285*a^2 + 7210368565303*a + 6470443499725
Resultado: True
------------------
i=13: q_i = 7950399097356*a^2 + 5148835888538*a + 2128139913306
x_i*Δ + m_i = 4058757332235*688778427917*a^2 + 2218026672466*a + 7422439420854 + 383179179825*a^2 +
3626669118578*a + 3243732715742 = 7950399097356*a^2 + 5148835888538*a + 2128139913306
Resultado: True
------------------
i=14: q_i = 3419223626241*a^2 + 2565334259105*a + 5490831904830
x_i*Δ + m_i = 1590256658158*688778427917*a^2 + 2218026672466*a + 7422439420854 + 4656220105022*a^2 +
8404807863090*a + 5410940372208 = 3419223626241*a^2 + 2565334259105*a + 5490831904830
Resultado: True
------------------
i=15: q_i = 6357175813261*a^2 + 7262623828517*a + 4124064324150
x_i*Δ + m_i = 1646743323258*688778427917*a^2 + 2218026672466*a + 7422439420854 + 7842764112681*a^2 +
5160639851123*a + 6329683808184 = 6357175813261*a^2 + 7262623828517*a + 4124064324150
```

```
Resultado: True
------------------
i=16: q_i = 3910199624738*a^2 + 2560211669155*a + 1714757257403
x_i*Δ + m_i = 5991288576178*688778427917*a^2 + 2218026672466*a + 7422439420854 + 1630037626632*a^2 +
8337365093622*a + 1647879084350 = 3910199624738*a^2 + 2560211669155*a + 1714757257403
Resultado: True
------------------
i=17: q_i = 2715710482859*a^2 + 8283618826057*a + 7432067567947
x_i*Δ + m_i = 177541916936*688778427917*a^2 + 2218026672466*a + 7422439420854 + 3644162492860*a^2 +
5233052112029*a + 6370359784728 = 2715710482859*a^2 + 8283618826057*a + 7432067567947
Resultado: True
------------------
i=18: q_i = 7077818254962*a^2 + 2942468445916*a + 8281189207012
x_i*Δ + m_i = 5643524405883*688778427917*a^2 + 2218026672466*a + 7422439420854 + 1692859130228*a^2 +
3150829886832*a + 5181362759752 = 7077818254962*a^2 + 2942468445916*a + 8281189207012
Resultado: True
------------------
i=19: q_i = 7498129295574*a^2 + 6742053367578*a + 4794087173838
x_i*Δ + m_i = 3484186328349*688778427917*a^2 + 2218026672466*a + 7422439420854 + 7252953282654*a^2 +
4326237963065*a + 3410363834805 = 7498129295574*a^2 + 6742053367578*a + 4794087173838
Resultado: True
------------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
```

```
For i=19, check: True
Verificação: 6372416211533*a^2 + 86294390276*a + 4765808213363 == 6372416211533*a^2 + 86294390276*a
+ 4765808213363
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 4


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 3211172037945*a^2 + 1909862879455*a + 5288660409881 (índice j = 5)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 6613189557163*a^2 + 3970939519524*a + 7786615603252
x_i*Δ + m_i = 7271547995351*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 987574127044*a^2 +
4278491619359*a + 5235397053601 = 6613189557163*a^2 + 3970939519524*a + 7786615603252
Resultado: True
------------------
i=1: q_i = 6842620285653*a^2 + 5866763757669*a + 5415080316271
x_i*Δ + m_i = 4925088477414*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4058971983669*a^2
+ 370170034482*a + 1622764015544 = 6842620285653*a^2 + 5866763757669*a + 5415080316271
Resultado: True
------------------
i=2: q_i = 1714564726972*a^2 + 1305995955490*a + 738116677290
x_i*Δ + m_i = 3253063993110*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 201577614869*a^2 +
1535054523590*a + 4193474585056 = 1714564726972*a^2 + 1305995955490*a + 738116677290
Resultado: True
------------------
i=3: q_i = 1402730320703*a^2 + 3128080871707*a + 7622477199143
x_i*Δ + m_i = 8756199863205*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1482009948934*a^2
+ 1885402111314*a + 8278819633567 = 1402730320703*a^2 + 3128080871707*a + 7622477199143
Resultado: True
------------------
i=4: q_i = 2832640496989*a^2 + 5549382910287*a + 3317147098886
x_i*Δ + m_i = 1461394958902*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3421756698620*a^2
+ 8092119605942*a + 8654494378872 = 2832640496989*a^2 + 5549382910287*a + 3317147098886
Resultado: True
------------------
i=5: q_i = 3214157226759*a^2 + 2611615015748*a + 7190248186302
x_i*Δ + m_i = 8244089003708*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4361136866254*a^2
+ 4750752706771*a + 6715179485391 = 3214157226759*a^2 + 2611615015748*a + 7190248186302
Resultado: True
------------------
i=6: q_i = 872758125476*a^2 + 1565223826451*a + 8326617615843
x_i*Δ + m_i = 6377035704846*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8583238080687*a^2
+ 5811736120684*a + 389220471162 = 872758125476*a^2 + 1565223826451*a + 8326617615843
Resultado: True
------------------
i=7: q_i = 7735047886143*a^2 + 2485740032296*a + 4521488674338
x_i*Δ + m_i = 6821138990438*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2600602156982*a^2
+ 7820543128582*a + 3389613441938 = 7735047886143*a^2 + 2485740032296*a + 4521488674338
Resultado: True
------------------
i=8: q_i = 5418987448010*a^2 + 6295826110942*a + 7769636863485
x_i*Δ + m_i = 5222266180217*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5983885956786*a^2
+ 7897020453786*a + 6140811374726 = 5418987448010*a^2 + 6295826110942*a + 7769636863485
Resultado: True
------------------
```

```
i=9: q_i = 2239028575901*a^2 + 7909861770138*a + 2250566231583
x_i*Δ + m_i = 2717994838566*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2041890761178*a^2
+ 352627158276*a + 4754983246504 = 2239028575901*a^2 + 7909861770138*a + 2250566231583
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
----------------------------FASE MU---------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=6 bem decifrada
------------------
i=0: q_i = 7031758693481*a^2 + 8230572978238*a + 2923938420543
x_i*Δ + m_i = 4577434244125*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4625301755968*a^2
+ 3390638014937*a + 26249385405 = 7031758693481*a^2 + 8230572978238*a + 2923938420543
Resultado: True
------------------
i=1: q_i = 1855211422061*a^2 + 1361144204289*a + 4630077254482
x_i*Δ + m_i = 8795015784975*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8030334948149*a^2
+ 884097763629*a + 2899354464899 = 1855211422061*a^2 + 1361144204289*a + 4630077254482
Resultado: True
------------------
i=2: q_i = 721630248103*a^2 + 7961612935279*a + 7603652085624
x_i*Δ + m_i = 4979047102823*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5159324694060*a^2
+ 5643319081466*a + 6126486103439 = 721630248103*a^2 + 7961612935279*a + 7603652085624
Resultado: True
------------------
i=3: q_i = 5041676325783*a^2 + 6737582270063*a + 694013842423
x_i*Δ + m_i = 375593837625*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 141163138158*a^2 +
1013725631130*a + 1631196553958 = 5041676325783*a^2 + 6737582270063*a + 694013842423
Resultado: True
------------------
i=4: q_i = 293179693242*a^2 + 3899933627966*a + 326733662586
x_i*Δ + m_i = 5278959248757*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3146917129717*a^2
+ 152281253859*a + 4141713465518 = 293179693242*a^2 + 3899933627966*a + 326733662586
Resultado: True
------------------
i=5: q_i = 1666113961392*a^2 + 69748788530*a + 2248530945962
x_i*Δ + m_i = 1018094938099*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2165034601358*a^2
+ 6281871373830*a + 8435660821135 = 1666113961392*a^2 + 69748788530*a + 2248530945962
Resultado: True
------------------
i=6: q_i = 5449816344637*a^2 + 4051321554702*a + 2445810242294
x_i*Δ + m_i = 5919611164908*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5111923969857*a^2
+ 2566859192767*a + 4643480904311 = 5449816344637*a^2 + 4051321554702*a + 2445810242294
Resultado: True
------------------
i=7: q_i = 7634691961958*a^2 + 7284163822438*a + 5407033900551
x_i*Δ + m_i = 7012678867648*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7119351130739*a^2
+ 7297105946095*a + 8386001551948 = 7634691961958*a^2 + 7284163822438*a + 5407033900551
Resultado: True
------------------
i=8: q_i = 3185769216994*a^2 + 6536378279147*a + 7601789907484
x_i*Δ + m_i = 3874505412695*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 732622970382*a^2 +
1708471026056*a + 763976162439 = 3185769216994*a^2 + 6536378279147*a + 7601789907484
Resultado: True
------------------
i=9: q_i = 1657260318870*a^2 + 6237780361884*a + 4770294650329
x_i*Δ + m_i = 7130946633693*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1102388135439*a^2
+ 6293186734916*a + 5155263999918 = 1657260318870*a^2 + 6237780361884*a + 4770294650329
Resultado: True
------------------
i=10: q_i = 3546712154169*a^2 + 7174235433081*a + 5225502659538
x_i*Δ + m_i = 3444420300587*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 89782230598*a^2 +
7710857420860*a + 2361305115137 = 3546712154169*a^2 + 7174235433081*a + 5225502659538
Resultado: True
------------------
i=11: q_i = 2344779946332*a^2 + 1902358894147*a + 7243856745681
x_i*Δ + m_i = 5727598551275*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4526698948276*a^2
+ 4054089951517*a + 3879634783256 = 2344779946332*a^2 + 1902358894147*a + 7243856745681
Resultado: True
------------------
i=12: q_i = 6032590002484*a^2 + 6906861691024*a + 6521125054724
x_i*Δ + m_i = 971269061061*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6636290442805*a^2 +
5241563615654*a + 5083184464369 = 6032590002484*a^2 + 6906861691024*a + 6521125054724
Resultado: True
------------------
```

```
i=13: q_i = 2302488426487*a^2 + 3800579527826*a + 159137610212
x_i*Δ + m_i = 1805823363445*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6846274471907*a^2
+ 8312332919446*a + 6968804320465 = 2302488426487*a^2 + 3800579527826*a + 159137610212
Resultado: True
------------------
i=14: q_i = 4902506068623*a^2 + 3465100223775*a + 4026792628518
x_i*Δ + m_i = 6396569499540*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 39705550254*a^2 +
5940225117423*a + 5604884502441 = 4902506068623*a^2 + 3465100223775*a + 4026792628518
Resultado: True
------------------
i=15: q_i = 3250848947851*a^2 + 7733982780079*a + 864948488005
x_i*Δ + m_i = 5800563482678*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6795213733539*a^2
+ 7383845552503*a + 3985477110876 = 3250848947851*a^2 + 7733982780079*a + 864948488005
Resultado: True
------------------
i=16: q_i = 8396488862602*a^2 + 8078603843403*a + 3088587770753
x_i*Δ + m_i = 6327666512226*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3703123237718*a^2
+ 3766757517703*a + 3486124518065 = 8396488862602*a^2 + 8078603843403*a + 3088587770753
Resultado: True
------------------
i=17: q_i = 7321726649893*a^2 + 1420726272313*a + 23368740457
x_i*Δ + m_i = 8284259307956*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7024733009013*a^2
+ 1777878535608*a + 459256470646 = 7321726649893*a^2 + 1420726272313*a + 23368740457
Resultado: True
------------------
i=18: q_i = 8485905958100*a^2 + 7152498625135*a + 1165201815875
x_i*Δ + m_i = 3142861807741*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8422813419946*a^2
+ 1911162707589*a + 3926983654431 = 8485905958100*a^2 + 7152498625135*a + 1165201815875
Resultado: True
------------------
i=19: q_i = 6593525575621*a^2 + 613682737538*a + 17206386128
x_i*Δ + m_i = 3236499109134*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6286713366948*a^2
+ 3820243473351*a + 2465107613222 = 6593525575621*a^2 + 613682737538*a + 17206386128
Resultado: True
------------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
```

```
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 7402648663890*a^2 + 8495813843173*a + 7575736961930 == 7402648663890*a^2 + 849581384317
3*a + 7575736961930
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 5


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 3211172037945*a^2 + 1909862879455*a + 5288660409881 (índice j = 5)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 431739081656*a^2 + 4480473252601*a + 2740358092967
x_i*Δ + m_i = 7271547995351*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3602216673774*a^2
+ 4788025352436*a + 189139543316 = 431739081656*a^2 + 4480473252601*a + 2740358092967
Resultado: True
------------------
i=1: q_i = 5489772210479*a^2 + 2899628823745*a + 6847955406699
x_i*Δ + m_i = 4925088477414*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2706123908495*a^2
+ 6199128122795*a + 3055639105972 = 5489772210479*a^2 + 2899628823745*a + 6847955406699
Resultado: True
------------------
i=2: q_i = 5668876350538*a^2 + 4339102640771*a + 5438030601948
x_i*Δ + m_i = 3253063993110*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4155889238435*a^2
+ 4568161208871*a + 97295487477 = 5668876350538*a^2 + 4339102640771*a + 5438030601948
Resultado: True
------------------
i=3: q_i = 8202997762968*a^2 + 5726150137370*a + 3670575551595
x_i*Δ + m_i = 8756199863205*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8282277391199*a^2
+ 4483471376977*a + 4326917986019 = 8202997762968*a^2 + 5726150137370*a + 3670575551595
Resultado: True
------------------
i=4: q_i = 5793737866077*a^2 + 3331425397335*a + 3171622292359
x_i*Δ + m_i = 1461394958902*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6382854067708*a^2
+ 5874162092990*a + 8508969572345 = 5793737866077*a^2 + 3331425397335*a + 3171622292359
Resultado: True
------------------
i=5: q_i = 6207573447273*a^2 + 1145235698640*a + 2708090829893
x_i*Δ + m_i = 8244089003708*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7354553086768*a^2
+ 3284373389663*a + 2233022128982 = 6207573447273*a^2 + 1145235698640*a + 2708090829893
Resultado: True
------------------
i=6: q_i = 6529798813457*a^2 + 182188822594*a + 8177891679840
x_i*Δ + m_i = 6377035704846*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5444185746431*a^2
```

```
+ 4428701116827*a + 240494535159 = 6529798813457*a^2 + 182188822594*a + 8177891679840
Resultado: True
------------------
i=7: q_i = 6186747966230*a^2 + 5886097595386*a + 1304753904460
x_i*Δ + m_i = 6821138990438*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1052302237069*a^2
+ 2424807669435*a + 172878672060 = 6186747966230*a^2 + 5886097595386*a + 1304753904460
Resultado: True
------------------
i=8: q_i = 4464887390496*a^2 + 5658128592084*a + 8687364790080
x_i*Δ + m_i = 5222266180217*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5029785899272*a^2
+ 7259322934928*a + 7058539301321 = 4464887390496*a^2 + 5658128592084*a + 8687364790080
Resultado: True
------------------
i=9: q_i = 4858623673595*a^2 + 3103884232874*a + 6724054008795
x_i*Δ + m_i = 2717994838566*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4661485858872*a^2
+ 4342742643249*a + 432378001479 = 4858623673595*a^2 + 3103884232874*a + 6724054008795
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
----------------------------FASE MU-------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=6 bem decifrada
------------------
i=0: q_i = 1349083542700*a^2 + 316989411602*a + 3358162569432
x_i*Δ + m_i = 6228527867904*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 833390668967*a^2 +
5339828206463*a + 6739649837150 = 1349083542700*a^2 + 316989411602*a + 3358162569432
Resultado: True
------------------
i=1: q_i = 2094674793727*a^2 + 3867698335657*a + 3893271950284
x_i*Δ + m_i = 5240612546353*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 26707278076*a^2 +
5193342385211*a + 5493483419590 = 2094674793727*a^2 + 3867698335657*a + 3893271950284
Resultado: True
------------------
i=2: q_i = 6762756754077*a^2 + 6671727357806*a + 6134098448273
x_i*Δ + m_i = 4066922391753*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4442073476491*a^2
+ 7432747928881*a + 8389110479321 = 6762756754077*a^2 + 6671727357806*a + 6134098448273
Resultado: True
------------------
i=3: q_i = 4410345830927*a^2 + 557004587876*a + 407352235400
x_i*Δ + m_i = 4630741549615*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5041906788301*a^2
+ 2422535004489*a + 2278173760132 = 4410345830927*a^2 + 557004587876*a + 407352235400
Resultado: True
------------------
i=4: q_i = 4371901515434*a^2 + 2558581354695*a + 7544819430711
x_i*Δ + m_i = 3789856060965*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2202418054215*a^2
+ 2672764770106*a + 1731825179039 = 4371901515434*a^2 + 2558581354695*a + 7544819430711
Resultado: True
------------------
i=5: q_i = 22070704938*a^2 + 3094341795389*a + 7117573665070
x_i*Δ + m_i = 3293941065416*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2159389467976*a^2
+ 929069098752*a + 6086520833110 = 22070704938*a^2 + 3094341795389*a + 7117573665070
Resultado: True
------------------
i=6: q_i = 3088567213017*a^2 + 4931870392306*a + 6350850346322
x_i*Δ + m_i = 4455857415217*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6430833432997*a^2
+ 6762677977535*a + 4494488582014 = 3088567213017*a^2 + 4931870392306*a + 6350850346322
Resultado: True
------------------
i=7: q_i = 5264779254543*a^2 + 6492816226747*a + 1444370489697
x_i*Δ + m_i = 1157564693196*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6352517330097*a^2
+ 5110248076287*a + 8435246706503 = 5264779254543*a^2 + 6492816226747*a + 1444370489697
Resultado: True
------------------
i=8: q_i = 7807225424115*a^2 + 2536592540259*a + 5796033961638
x_i*Δ + m_i = 1007361451387*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4588704842242*a^2
+ 2148034869843*a + 2632101480248 = 7807225424115*a^2 + 2536592540259*a + 5796033961638
Resultado: True
------------------
i=9: q_i = 4372121513785*a^2 + 573703726661*a + 4564062198568
x_i*Δ + m_i = 7147896610285*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2866948513197*a^2
+ 8236732730525*a + 2302027742083 = 4372121513785*a^2 + 573703726661*a + 4564062198568
Resultado: True
------------------
i=10: q_i = 813522148166*a^2 + 6176555565034*a + 8773057759602
x_i*Δ + m_i = 6022080015422*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1914657530747*a^2
```

+ 4587063920376*a + 6468503826755 = 813522148166*a^2 + 6176555565034*a + 8773057759602
Resultado: True
----
i=11: q_i = 6027546359192*a^2 + 7256525540467*a + 6147266577305
x_i*Δ + m_i = 4268245730705*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6783504826530*a^2
+ 3377429760919*a + 5426588480540 = 6027546359192*a^2 + 7256525540467*a + 6147266577305
Resultado: True
----
i=12: q_i = 2864932793506*a^2 + 5020216144886*a + 1730064202218
x_i*Δ + m_i = 7299938739978*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8406908785509*a^2
+ 7351485959439*a + 8099107843439 = 2864932793506*a^2 + 5020216144886*a + 1730064202218
Resultado: True
----
i=13: q_i = 1019181694016*a^2 + 5737645079361*a + 7201662085255
x_i*Δ + m_i = 5341317778048*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1186533486636*a^2
+ 7569502863051*a + 1822826839544 = 1019181694016*a^2 + 5737645079361*a + 7201662085255
Resultado: True
----
i=14: q_i = 1207886763996*a^2 + 6205097458608*a + 1823801012173
x_i*Δ + m_i = 5803200192450*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2928733975620*a^2
+ 4505936092529*a + 3093490530216 = 1207886763996*a^2 + 6205097458608*a + 1823801012173
Resultado: True
----
i=15: q_i = 266552336403*a^2 + 8506445518468*a + 6547659946489
x_i*Δ + m_i = 7326431538217*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1972951379717*a^2
+ 6282455916828*a + 7370518624229 = 266552336403*a^2 + 8506445518468*a + 6547659946489
Resultado: True
----
i=16: q_i = 7597713184094*a^2 + 2745687384129*a + 8222596029255
x_i*Δ + m_i = 4346817771389*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5810559170574*a^2
+ 8197736525264*a + 1351034724014 = 7597713184094*a^2 + 2745687384129*a + 8222596029255
Resultado: True
----
i=17: q_i = 1328452284206*a^2 + 7777620355897*a + 4105769051091
x_i*Δ + m_i = 3436200413947*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3473824523062*a^2
+ 1378426370007*a + 5963083853393 = 1328452284206*a^2 + 7777620355897*a + 4105769051091
Resultado: True
----
i=18: q_i = 7802546659046*a^2 + 7986369449690*a + 4770004580795
x_i*Δ + m_i = 2961539224120*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7916745956668*a^2
+ 6223644046361*a + 8785930840216 = 7802546659046*a^2 + 7986369449690*a + 4770004580795
Resultado: True
----
i=19: q_i = 6650651075253*a^2 + 6071480348351*a + 7194042472952
x_i*Δ + m_i = 7180171557182*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3582177353284*a^2
+ 1227150369981*a + 3679622427069 = 6650651075253*a^2 + 6071480348351*a + 7194042472952
Resultado: True

Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16

```
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 1320135569594*a^2 + 5731324763482*a + 1289811979883 == 1320135569594*a^2 + 573132476348
2*a + 1289811979883
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 6


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 351186518504*a^2 + 710356990350*a + 6130922975946 (índice j = 6)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 5762872695403*a^2 + 2353846499763*a + 8349566260641
x_i*Δ + m_i = 7271547995351*351186518504*a^2 + 710356990350*a + 6130922975946 + 5139395733795*a^2 +
5645880293337*a + 7492456286140 = 5762872695403*a^2 + 2353846499763*a + 8349566260641
Resultado: True
------------------
i=1: q_i = 5964564768905*a^2 + 6630030128155*a + 4987757252402
x_i*Δ + m_i = 4925088477414*351186518504*a^2 + 710356990350*a + 6130922975946 + 7920429551011*a^2 +
5593123973658*a + 4220337336082 = 5964564768905*a^2 + 6630030128155*a + 4987757252402
Resultado: True
------------------
i=2: q_i = 7664899482745*a^2 + 4767546248763*a + 5514524738978
x_i*Δ + m_i = 3253063993110*351186518504*a^2 + 710356990350*a + 6130922975946 + 5254842322468*a^2 +
2011138990103*a + 5137065529993 = 7664899482745*a^2 + 4767546248763*a + 5514524738978
Resultado: True
------------------
i=3: q_i = 2350552953717*a^2 + 3918538428379*a + 6752186825061
x_i*Δ + m_i = 8756199863205*351186518504*a^2 + 710356990350*a + 6130922975946 + 6685979817124*a^2 +
3209454805820*a + 4881831561922 = 2350552953717*a^2 + 3918538428379*a + 6752186825061
Resultado: True
```

```
------------------
i=4: q_i = 3644793747901*a^2 + 2983465259829*a + 6628406491780
x_i*Δ + m_i = 1461394958902*351186518504*a^2 + 710356990350*a + 6130922975946 + 324864459743*a^2 + 3
37819426312*a + 741660100413 = 3644793747901*a^2 + 2983465259829*a + 6628406491780
Resultado: True
------------------
i=5: q_i = 2550350774181*a^2 + 8613019430448*a + 7530419477217
x_i*Δ + m_i = 8244089003708*351186518504*a^2 + 710356990350*a + 6130922975946 + 1417576085568*a^2 +
3682624903159*a + 7343686422400 = 2550350774181*a^2 + 8613019430448*a + 7530419477217
Resultado: True
------------------
i=6: q_i = 3398529091814*a^2 + 6870161247590*a + 1240701408323
x_i*Δ + m_i = 6377035704846*351186518504*a^2 + 710356990350*a + 6130922975946 + 2719254390189*a^2 +
7000104193208*a + 4897971660589 = 3398529091814*a^2 + 6870161247590*a + 1240701408323
Resultado: True
------------------
i=7: q_i = 8612814144658*a^2 + 7751202304633*a + 4487680362985
x_i*Δ + m_i = 6821138990438*351186518504*a^2 + 710356990350*a + 6130922975946 + 816662889475*a^2 + 7
613000919315*a + 6145280701205 = 8612814144658*a^2 + 7751202304633*a + 4487680362985
Resultado: True
------------------
i=8: q_i = 4031825369289*a^2 + 3207389992064*a + 7109999739821
x_i*Δ + m_i = 5222266180217*351186518504*a^2 + 710356990350*a + 6130922975946 + 1526983171957*a^2 +
4318314476761*a + 7934532085971 = 4031825369289*a^2 + 3207389992064*a + 7109999739821
Resultado: True
------------------
i=9: q_i = 6669764508205*a^2 + 6212424091505*a + 989919290142
x_i*Δ + m_i = 2717994838566*351186518504*a^2 + 710356990350*a + 6130922975946 + 6484924491154*a^2 +
6102635823539*a + 4911120259403 = 6669764508205*a^2 + 6212424091505*a + 989919290142
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
-----------------------------FASE MU--------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=5 bem decifrada
------------------
i=0: q_i = 7023452025515*a^2 + 5814038240125*a + 7104044297941
x_i*Δ + m_i = 5261043234741*351186518504*a^2 + 710356990350*a + 6130922975946 + 2649954493578*a^2 +
4071521944449*a + 5787372162609 = 7023452025515*a^2 + 5814038240125*a + 7104044297941
Resultado: True
------------------
i=1: q_i = 6700669494737*a^2 + 3674517865051*a + 984212751887
x_i*Δ + m_i = 1070991160790*351186518504*a^2 + 710356990350*a + 6130922975946 + 1876092880716*a^2 +
4209932802274*a + 2211373457610 = 6700669494737*a^2 + 3674517865051*a + 984212751887
Resultado: True
------------------
i=2: q_i = 5049622510914*a^2 + 5864794242625*a + 4264624479595
x_i*Δ + m_i = 896213785402*351186518504*a^2 + 710356990350*a + 6130922975946 + 2952410915159*a^2 + 8
526831079352*a + 5804877979219 = 5049622510914*a^2 + 5864794242625*a + 4264624479595
Resultado: True
------------------
i=3: q_i = 8648079370194*a^2 + 8050403343171*a + 5075880431040
x_i*Δ + m_i = 5455929099789*351186518504*a^2 + 710356990350*a + 6130922975946 + 7284192434063*a^2 +
6742393987975*a + 2980876042174 = 8648079370194*a^2 + 8050403343171*a + 5075880431040
Resultado: True
------------------
i=4: q_i = 3088108090743*a^2 + 8058495171495*a + 1366533893661
x_i*Δ + m_i = 5812748423003*351186518504*a^2 + 710356990350*a + 6130922975946 + 4233309962716*a^2 +
5859683649899*a + 5492851940397 = 3088108090743*a^2 + 8058495171495*a + 1366533893661
Resultado: True
------------------
i=5: q_i = 159015007143*a^2 + 1245910167625*a + 1677654260479
x_i*Δ + m_i = 7577241319117*351186518504*a^2 + 710356990350*a + 6130922975946 + 6706656609286*a^2 +
2210336668063*a + 7195246250210 = 159015007143*a^2 + 1245910167625*a + 1677654260479
Resultado: True
------------------
i=6: q_i = 3760151909388*a^2 + 1715529238505*a + 1114422316591
x_i*Δ + m_i = 4645413746845*351186518504*a^2 + 710356990350*a + 6130922975946 + 8222497995546*a^2 +
3873624272661*a + 7329165592220 = 3760151909388*a^2 + 1715529238505*a + 1114422316591
Resultado: True
------------------
i=7: q_i = 7441501289298*a^2 + 962660440131*a + 4608743622244
x_i*Δ + m_i = 1971586716243*351186518504*a^2 + 710356990350*a + 6130922975946 + 983926319647*a^2 + 5
002301797000*a + 8316604914615 = 7441501289298*a^2 + 962660440131*a + 4608743622244
Resultado: True
```

```
------------------
i=8: q_i = 2200120070812*a^2 + 760968405706*a + 994304044581
x_i*Δ + m_i = 5007217982653*351186518504*a^2 + 710356990350*a + 6130922975946 + 2884671214496*a^2 +
1347951425341*a + 3606804165213 = 2200120070812*a^2 + 760968405706*a + 994304044581
Resultado: True
------------------
i=9: q_i = 2215529627720*a^2 + 6358448465367*a + 2273022902873
x_i*Δ + m_i = 696856146997*351186518504*a^2 + 710356990350*a + 6130922975946 + 2093726778799*a^2 + 4
018532041162*a + 7078503306571 = 2215529627720*a^2 + 6358448465367*a + 2273022902873
Resultado: True
------------------
i=10: q_i = 3821406259235*a^2 + 1683174593878*a + 5160196005321
x_i*Δ + m_i = 1576029517539*351186518504*a^2 + 710356990350*a + 6130922975946 + 4743226864644*a^2 +
2199054088493*a + 2944473645512 = 3821406259235*a^2 + 1683174593878*a + 5160196005321
Resultado: True
------------------
i=11: q_i = 3449873846262*a^2 + 2821317546590*a + 8198045552065
x_i*Δ + m_i = 457740343333*351186518504*a^2 + 710356990350*a + 6130922975946 + 2381919290358*a^2 + 7
937590831401*a + 6624499540419 = 3449873846262*a^2 + 2821317546590*a + 8198045552065
Resultado: True
------------------
i=12: q_i = 4067354824129*a^2 + 7787522416802*a + 8262999718399
x_i*Δ + m_i = 7657245770691*351186518504*a^2 + 710356990350*a + 6130922975946 + 6360847219609*a^2 +
7287017260344*a + 2761980589111 = 4067354824129*a^2 + 7787522416802*a + 8262999718399
Resultado: True
------------------
i=13: q_i = 7431102438291*a^2 + 7277975779244*a + 6904177917817
x_i*Δ + m_i = 7445285612834*351186518504*a^2 + 710356990350*a + 6130922975946 + 7746516879227*a^2 +
2421822814561*a + 3806025933327 = 7431102438291*a^2 + 7277975779244*a + 6904177917817
Resultado: True
------------------
i=14: q_i = 5533344481326*a^2 + 5874068000297*a + 1877070623478
x_i*Δ + m_i = 6434066058338*351186518504*a^2 + 710356990350*a + 6130922975946 + 5409242892907*a^2 +
5174607459131*a + 8468889481870 = 5533344481326*a^2 + 5874068000297*a + 1877070623478
Resultado: True
------------------
i=15: q_i = 7785697801548*a^2 + 6362025152607*a + 1385683087326
x_i*Δ + m_i = 4585925616724*351186518504*a^2 + 710356990350*a + 6130922975946 + 590549289895*a^2 + 5
753310359242*a + 725375537752 = 7785697801548*a^2 + 6362025152607*a + 1385683087326
Resultado: True
------------------
i=16: q_i = 2806943613406*a^2 + 7212356106072*a + 6049345137855
x_i*Δ + m_i = 226698822336*351186518504*a^2 + 710356990350*a + 6130922975946 + 4807668374646*a^2 + 6
331009017084*a + 5773390906284 = 2806943613406*a^2 + 7212356106072*a + 6049345137855
Resultado: True
------------------
i=17: q_i = 8275245737221*a^2 + 6831051684040*a + 5939619370192
x_i*Δ + m_i = 8009413680355*351186518504*a^2 + 710356990350*a + 6130922975946 + 8001023992398*a^2 +
7904544568555*a + 7512991863415 = 8275245737221*a^2 + 6831051684040*a + 5939619370192
Resultado: True
------------------
i=18: q_i = 4056162896170*a^2 + 4071012539895*a + 372835277653
x_i*Δ + m_i = 7586406339394*351186518504*a^2 + 710356990350*a + 6130922975946 + 62482840896*a^2 + 42
37989410105*a + 8489948514638 = 4056162896170*a^2 + 4071012539895*a + 372835277653
Resultado: True
------------------
i=19: q_i = 1600049615401*a^2 + 4133745081003*a + 3132763817888
x_i*Δ + m_i = 6451622339250*351186518504*a^2 + 710356990350*a + 6130922975946 + 7014125026389*a^2 +
8724926952639*a + 7797149538510 = 1600049615401*a^2 + 4133745081003*a + 3132763817888
Resultado: True
------------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
```

```
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 1638717883861*a^2 + 1500751470029*a + 3928577626602 == 1638717883861*a^2 + 150075147002
9*a + 3928577626602
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 7


--------------------------------------------------------------------------------
-----------------------------FASE W---------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 3296170799356*a^2 + 8684033073275*a + 993307392858 (índice j = 3)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 7568177633555*a^2 + 5814113041060*a + 1829956660566
x_i*Δ + m_i = 7271547995351*3296170799356*a^2 + 8684033073275*a + 993307392858 + 714585877726*a^2 +
4781943594908*a + 2614216391039 = 7568177633555*a^2 + 5814113041060*a + 1829956660566
Resultado: True
------------------
i=1: q_i = 6023808854310*a^2 + 4500692483564*a + 1168643672849
```

```
x_i*Δ + m_i = 4925088477414*3296170799356*a^2 + 8684033073275*a + 993307392858 + 2471529009160*a^2 +
7795900126082*a + 3720365956279 = 6023808854310*a^2 + 4500692483564*a + 1168643672849
Resultado: True
------------------
i=2: q_i = 6210172306201*a^2 + 8036397699800*a + 5735758046284
x_i*Δ + m_i = 3253063993110*3296170799356*a^2 + 8684033073275*a + 993307392858 + 5867439842370*a^2 +
4640425803438*a + 7276505402006 = 6210172306201*a^2 + 8036397699800*a + 5735758046284
Resultado: True
------------------
i=3: q_i = 6986797606503*a^2 + 1315582746948*a + 6958776502913
x_i*Δ + m_i = 8756199863205*3296170799356*a^2 + 8684033073275*a + 993307392858 + 5691526023804*a^2 +
1118830088115*a + 2891211103774 = 6986797606503*a^2 + 1315582746948*a + 6958776502913
Resultado: True
------------------
i=4: q_i = 762161783831*a^2 + 2890058191748*a + 2173888696989
x_i*Δ + m_i = 1461394958902*3296170799356*a^2 + 8684033073275*a + 993307392858 + 1633136214340*a^2 +
4422194134717*a + 5816878063643 = 762161783831*a^2 + 2890058191748*a + 2173888696989
Resultado: True
------------------
i=5: q_i = 1520180248903*a^2 + 1797529345633*a + 1583595903965
x_i*Δ + m_i = 8244089003708*3296170799356*a^2 + 8684033073275*a + 993307392858 + 3311125448207*a^2 +
6491214901107*a + 6908733032438 = 1520180248903*a^2 + 1797529345633*a + 1583595903965
Resultado: True
------------------
i=6: q_i = 2501786251521*a^2 + 7611876494257*a + 8365260532677
x_i*Δ + m_i = 6377035704846*3296170799356*a^2 + 8684033073275*a + 993307392858 + 4292883693477*a^2 +
2202549923365*a + 4161537430447 = 2501786251521*a^2 + 7611876494257*a + 8365260532677
Resultado: True
------------------
i=7: q_i = 7571909768382*a^2 + 4569395410451*a + 558540153066
x_i*Δ + m_i = 6821138990438*3296170799356*a^2 + 8684033073275*a + 993307392858 + 6175379459624*a^2 +
8229234816757*a + 8070797951080 = 7571909768382*a^2 + 4569395410451*a + 558540153066
Resultado: True
------------------
i=8: q_i = 1158295719026*a^2 + 2171888172492*a + 7968160054192
x_i*Δ + m_i = 5222266180217*3296170799356*a^2 + 8684033073275*a + 993307392858 + 7720502376774*a^2 +
4028969220438*a + 460583460950 = 1158295719026*a^2 + 2171888172492*a + 7968160054192
Resultado: True
------------------
i=9: q_i = 1463938748403*a^2 + 1485833611248*a + 2390139779942
x_i*Δ + m_i = 2717994838566*3296170799356*a^2 + 8684033073275*a + 993307392858 + 5329824044820*a^2 +
2605515182885*a + 6139212438857 = 1463938748403*a^2 + 1485833611248*a + 2390139779942
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
-----------------------------FASE MU--------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=5 bem decifrada
Mensagem msg=6 bem decifrada
------------------
i=0: q_i = 8465388687443*a^2 + 3101735979859*a + 3585670972971
x_i*Δ + m_i = 1228887430750*3296170799356*a^2 + 8684033073275*a + 993307392858 + 2532277543098*a^2 +
1909973411695*a + 7834266081257 = 8465388687443*a^2 + 3101735979859*a + 3585670972971
Resultado: True
------------------
i=1: q_i = 6477399591447*a^2 + 1191659433328*a + 4812922538110
x_i*Δ + m_i = 2839749382631*3296170799356*a^2 + 8684033073275*a + 993307392858 + 864217896219*a^2 +
4885729777250*a + 7337788215034 = 6477399591447*a^2 + 1191659433328*a + 4812922538110
Resultado: True
------------------
i=2: q_i = 4103239553289*a^2 + 7652660278774*a + 5247699500148
x_i*Δ + m_i = 2813373055646*3296170799356*a^2 + 8684033073275*a + 993307392858 + 4914639583498*a^2 +
7319792579586*a + 8450056656916 = 4103239553289*a^2 + 7652660278774*a + 5247699500148
Resultado: True
------------------
i=3: q_i = 3717151665142*a^2 + 5041631363197*a + 7069872319758
x_i*Δ + m_i = 305440752586*3296170799356*a^2 + 8684033073275*a + 993307392858 + 3874031677031*a^2 +
3852292611379*a + 398358795977 = 3717151665142*a^2 + 5041631363197*a + 7069872319758
Resultado: True
------------------
i=4: q_i = 4973628063967*a^2 + 5611622287387*a + 3003817178663
x_i*Δ + m_i = 828884852262*3296170799356*a^2 + 8684033073275*a + 993307392858 + 306062484511*a^2 + 1
365167367821*a + 6703868295643 = 4973628063967*a^2 + 5611622287387*a + 3003817178663
Resultado: True
------------------
i=5: q_i = 5562427604485*a^2 + 375868107741*a + 3028213672904
```

```
x_i*Δ + m_i = 5633144903464*3296170799356*a^2 + 8684033073275*a + 993307392858 + 1585620693638*a^2 +
3871252810257*a + 3535988440025 = 5562427604485*a^2 + 375868107741*a + 3028213672904
Resultado: True
------------------
i=6: q_i = 5182724376846*a^2 + 3514698676676*a + 631927590766
x_i*Δ + m_i = 3168800472072*3296170799356*a^2 + 8684033073275*a + 993307392858 + 80704425998*a^2 + 5
574573928356*a + 5354719329610 = 5182724376846*a^2 + 3514698676676*a + 631927590766
Resultado: True
------------------
i=7: q_i = 2496617758417*a^2 + 2359193666388*a + 1872882125285
x_i*Δ + m_i = 4818605213863*3296170799356*a^2 + 8684033073275*a + 993307392858 + 4044027337239*a^2 +
3433915236695*a + 340905235788 = 2496617758417*a^2 + 2359193666388*a + 1872882125285
Resultado: True
------------------
i=8: q_i = 4842597765617*a^2 + 144749784403*a + 8622762983922
x_i*Δ + m_i = 591266192504*3296170799356*a^2 + 8684033073275*a + 993307392858 + 210040947334*a^2 + 1
383957286047*a + 4219557148346 = 4842597765617*a^2 + 144749784403*a + 8622762983922
Resultado: True
------------------
i=9: q_i = 1067818189332*a^2 + 2978203711564*a + 1844817420847
x_i*Δ + m_i = 5821412955277*3296170799356*a^2 + 8684033073275*a + 993307392858 + 3876268701245*a^2 +
1907941951855*a + 2487380416978 = 1067818189332*a^2 + 2978203711564*a + 1844817420847
Resultado: True
------------------
i=10: q_i = 4021776075934*a^2 + 2597916054801*a + 8787606364248
x_i*Δ + m_i = 7920388152222*3296170799356*a^2 + 8684033073275*a + 993307392858 + 3725963426053*a^2 +
833699625651*a + 3741511333439 = 4021776075934*a^2 + 2597916054801*a + 8787606364248
Resultado: True
------------------
i=11: q_i = 4623483366199*a^2 + 304082888040*a + 2046295272125
x_i*Δ + m_i = 8124417201052*3296170799356*a^2 + 8684033073275*a + 993307392858 + 8005483499927*a^2 +
7747438286805*a + 7217947638526 = 4623483366199*a^2 + 304082888040*a + 2046295272125
Resultado: True
------------------
i=12: q_i = 6758372856210*a^2 + 6414954599098*a + 6266719894626
x_i*Δ + m_i = 4600354063374*3296170799356*a^2 + 8684033073275*a + 993307392858 + 1737376256335*a^2 +
7775547582884*a + 3367610831886 = 6758372856210*a^2 + 6414954599098*a + 6266719894626
Resultado: True
------------------
i=13: q_i = 2826104052748*a^2 + 3559407245680*a + 4977226921448
x_i*Δ + m_i = 8418457325413*3296170799356*a^2 + 8684033073275*a + 993307392858 + 5342776153861*a^2 +
5620841535738*a + 345046136580 = 2826104052748*a^2 + 3559407245680*a + 4977226921448
Resultado: True
------------------
i=14: q_i = 2840575746240*a^2 + 656082791456*a + 3016075293830
x_i*Δ + m_i = 1924802818578*3296170799356*a^2 + 8684033073275*a + 993307392858 + 7865788755639*a^2 +
6309762813308*a + 611657445031 = 2840575746240*a^2 + 656082791456*a + 3016075293830
Resultado: True
------------------
i=15: q_i = 1773383297100*a^2 + 7007652434033*a + 580837232856
x_i*Δ + m_i = 2500746917935*3296170799356*a^2 + 8684033073275*a + 993307392858 + 3958460602040*a^2 +
6546954297852*a + 786864704871 = 1773383297100*a^2 + 7007652434033*a + 580837232856
Resultado: True
------------------
i=16: q_i = 2042485616338*a^2 + 56382417360*a + 3658010167723
x_i*Δ + m_i = 615790774200*3296170799356*a^2 + 8684033073275*a + 993307392858 + 5257657861613*a^2 +
6987459663268*a + 5313612326137 = 2042485616338*a^2 + 56382417360*a + 3658010167723
Resultado: True
------------------
i=17: q_i = 3977737850002*a^2 + 1924617414590*a + 3870606176764
x_i*Δ + m_i = 8247905493892*3296170799356*a^2 + 8684033073275*a + 993307392858 + 4626847730525*a^2 +
1000015602846*a + 5944116427925 = 3977737850002*a^2 + 1924617414590*a + 3870606176764
Resultado: True
------------------
i=18: q_i = 1478584071351*a^2 + 1722026550073*a + 7775052796439
x_i*Δ + m_i = 6362033192427*3296170799356*a^2 + 8684033073275*a + 993307392858 + 6720536978760*a^2 +
4595715980202*a + 4063894524680 = 1478584071351*a^2 + 1722026550073*a + 7775052796439
Resultado: True
------------------
i=19: q_i = 8093728920260*a^2 + 8178087095479*a + 6317031729553
x_i*Δ + m_i = 3978156707553*3296170799356*a^2 + 8684033073275*a + 993307392858 + 4066781150069*a^2 +
2276255341459*a + 39914033272 = 8093728920260*a^2 + 8178087095479*a + 6317031729553
Resultado: True
------------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
```

```
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 2247847504607*a^2 + 3522480097175*a + 5440114423278 == 2247847504607*a^2 + 352248009717
5*a + 5440114423278
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 8


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
```

```
--------------------------------------------------------------------------------
------------------
Δ = 1283704123849*a^2 + 8161608311351*a + 6144617414335 (índice j = 1)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 3330963184063*a^2 + 6085604680711*a + 5870804082457
x_i*Δ + m_i = 7271547995351*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 1308951024917*a^2
+ 4973480389901*a + 2679019918080 = 3330963184063*a^2 + 6085604680711*a + 5870804082457
Resultado: True
------------------
i=1: q_i = 8757185497405*a^2 + 1494589622750*a + 2630699869382
x_i*Δ + m_i = 4925088477414*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 1673676577201*a^2
+ 7650788643624*a + 2358767966582 = 8757185497405*a^2 + 1494589622750*a + 2630699869382
Resultado: True
------------------
i=2: q_i = 3168815505049*a^2 + 4059660724373*a + 7591555795565
x_i*Δ + m_i = 3253063993110*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 75201258126*a^2 +
4512991204857*a + 567717919968 = 3168815505049*a^2 + 4059660724373*a + 7591555795565
Resultado: True
------------------
i=3: q_i = 4140729412127*a^2 + 727871185006*a + 8781103590973
x_i*Δ + m_i = 8756199863205*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 8523872943306*a^2
+ 1161513188028*a + 2553444755636 = 4140729412127*a^2 + 727871185006*a + 8781103590973
Resultado: True
------------------
i=4: q_i = 4999910998514*a^2 + 2349576588261*a + 4971343235685
x_i*Δ + m_i = 1461394958902*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 5273330476235*a^2
+ 5308129740814*a + 4280138277589 = 4999910998514*a^2 + 2349576588261*a + 4971343235685
Resultado: True
------------------
i=5: q_i = 1745185240300*a^2 + 6170161904088*a + 531756556737
x_i*Δ + m_i = 8244089003708*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 6645920226766*a^2
+ 4236729786713*a + 2932539284715 = 1745185240300*a^2 + 6170161904088*a + 531756556737
Resultado: True
------------------
i=6: q_i = 899730102862*a^2 + 3780821555864*a + 2642806943099
x_i*Δ + m_i = 6377035704846*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 2112104845476*a^2
+ 8020912174091*a + 8102474900466 = 899730102862*a^2 + 3780821555864*a + 2642806943099
Resultado: True
------------------
i=7: q_i = 7541469860494*a^2 + 7510500382039*a + 593304815180
x_i*Δ + m_i = 6821138990438*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4459282470032*a^2
+ 6634306350506*a + 1408206723955 = 7541469860494*a^2 + 7510500382039*a + 593304815180
Resultado: True
------------------
i=8: q_i = 609990905187*a^2 + 634812048746*a + 2379758615663
x_i*Δ + m_i = 5222266180217*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 6691396505883*a^2
+ 6847074296196*a + 6015982938742 = 609990905187*a^2 + 634812048746*a + 2379758615663
Resultado: True
------------------
i=9: q_i = 533788524662*a^2 + 1889537852859*a + 7070788655219
x_i*Δ + m_i = 2717994838566*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 3656708857018*a^2
+ 6030060342452*a + 7614889999657 = 533788524662*a^2 + 1889537852859*a + 7070788655219
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
----------------------------FASE MU---------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=5 bem decifrada
Mensagem msg=6 bem decifrada
------------------
i=0: q_i = 3807890463221*a^2 + 4818819723987*a + 427531593373
x_i*Δ + m_i = 6160012174794*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4643285656758*a^2
+ 2988781880939*a + 5213187693340 = 3807890463221*a^2 + 4818819723987*a + 427531593373
Resultado: True
------------------
i=1: q_i = 617438846588*a^2 + 1910987025440*a + 2739699720957
x_i*Δ + m_i = 769695146418*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 349220551334*a^2 +
3543370770218*a + 7875124422415 = 617438846588*a^2 + 1910987025440*a + 2739699720957
Resultado: True
------------------
i=2: q_i = 6811390200661*a^2 + 1899696060151*a + 5250391469217
x_i*Δ + m_i = 8044260587024*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 1681433823990*a^2
+ 2408109431725*a + 8124847354104 = 6811390200661*a^2 + 1899696060151*a + 5250391469217
```

Resultado: True
---
i=3: q_i = 2996769932004*a^2 + 6245655306467*a + 6642586072533
x_i*Δ + m_i = 4198144840061*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 1685453508993*a^2
+ 4128044099420*a + 8520695065436 = 2996769932004*a^2 + 6245655306467*a + 6642586072533
Resultado: True
---
i=4: q_i = 8072803677297*a^2 + 7713060464505*a + 2757939686485
x_i*Δ + m_i = 10051151555*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4022386675817*a^2 +
1201818632846*a + 350799862339 = 8072803677297*a^2 + 7713060464505*a + 2757939686485
Resultado: True
---
i=5: q_i = 3826407936377*a^2 + 1902622324694*a + 1916475187354
x_i*Δ + m_i = 87763626770*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4910951433524*a^2 +
7814229253029*a + 3515289655129 = 3826407936377*a^2 + 1902622324694*a + 1916475187354
Resultado: True
---
i=6: q_i = 5553482867454*a^2 + 5679098527228*a + 231708768858
x_i*Δ + m_i = 6146233024212*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 3974276661185*a^2
+ 2895274412935*a + 883084792161 = 5553482867454*a^2 + 5679098527228*a + 231708768858
Resultado: True
---
i=7: q_i = 6852489057334*a^2 + 1831493411649*a + 6553906898928
x_i*Δ + m_i = 1009189668634*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 6700252768762*a^2
+ 6794874759843*a + 8285228928002 = 6852489057334*a^2 + 1831493411649*a + 6553906898928
Resultado: True
---
i=8: q_i = 7672587693593*a^2 + 4635329871773*a + 6322576782970
x_i*Δ + m_i = 2526767551980*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4937595734218*a^2
+ 2288845978874*a + 6422063946387 = 7672587693593*a^2 + 4635329871773*a + 6322576782970
Resultado: True
---
i=9: q_i = 3745682769520*a^2 + 271620876596*a + 468666454496
x_i*Δ + m_i = 785643342601*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4225791690731*a^2 +
610042380424*a + 4663045824746 = 3745682769520*a^2 + 271620876596*a + 468666454496
Resultado: True
---
i=10: q_i = 3991203945468*a^2 + 410188683071*a + 7268523254592
x_i*Δ + m_i = 6218975122024*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 2894687566602*a^2
+ 819365526853*a + 2853988418154 = 3991203945468*a^2 + 410188683071*a + 7268523254592
Resultado: True
---
i=11: q_i = 3325649932612*a^2 + 2568664628149*a + 7867941155257
x_i*Δ + m_i = 7710964624685*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 6847517664641*a^2
+ 1342502757283*a + 5503534463790 = 3325649932612*a^2 + 2568664628149*a + 7867941155257
Resultado: True
---
i=12: q_i = 736010458184*a^2 + 7761334219271*a + 1571479649756
x_i*Δ + m_i = 2481067804118*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 7290711524669*a^2
+ 3352969000217*a + 360513102383 = 736010458184*a^2 + 7761334219271*a + 1571479649756
Resultado: True
---
i=13: q_i = 3236673041849*a^2 + 3395002061444*a + 7525020532438
x_i*Δ + m_i = 5126736949027*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 6719774823739*a^2
+ 7375524368849*a + 7725816915609 = 3236673041849*a^2 + 3395002061444*a + 7525020532438
Resultado: True
---
i=14: q_i = 7693657875033*a^2 + 7048412864812*a + 5696847082335
x_i*Δ + m_i = 7491992195771*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 4954343147379*a^2
+ 1469099789344*a + 4807123966776 = 7693657875033*a^2 + 7048412864812*a + 5696847082335
Resultado: True
---
i=15: q_i = 5933200766298*a^2 + 5771214231471*a + 8741943622067
x_i*Δ + m_i = 3371807199138*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 1572915662942*a^2
+ 2031999038714*a + 1209794747102 = 5933200766298*a^2 + 5771214231471*a + 8741943622067
Resultado: True
---
i=16: q_i = 4718143709050*a^2 + 8685174446005*a + 8351706064377
x_i*Δ + m_i = 2194971722495*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 7443873585663*a^2
+ 1471232634822*a + 2893982221217 = 4718143709050*a^2 + 8685174446005*a + 8351706064377
Resultado: True
---
i=17: q_i = 5586522462703*a^2 + 2790430236249*a + 7232115042213
x_i*Δ + m_i = 3881431048541*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 7173341813381*a^2
+ 3801086886827*a + 6866262364810 = 5586522462703*a^2 + 2790430236249*a + 7232115042213
Resultado: True
---
i=18: q_i = 6832297771842*a^2 + 6730917264540*a + 3070042742968
x_i*Δ + m_i = 7109161068019*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 3081375023045*a^2
+ 6017354280184*a + 8072084234125 = 6832297771842*a^2 + 6730917264540*a + 3070042742968
Resultado: True
---
i=19: q_i = 3846047982526*a^2 + 2178095433898*a + 7809484442286

```
x_i*Δ + m_i = 3244350821527*1283704123849*a^2 + 8161608311351*a + 6144617414335 + 2927956280931*a^2
+ 1181464511719*a + 7052302477025 = 3846047982526*a^2 + 2178095433898*a + 7809484442286
Resultado: True
-----------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 4031184700025*a^2 + 1349462913506*a + 3050269912819 == 4031184700025*a^2 + 134946291350
6*a + 3050269912819
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
```

```
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 9


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 3211172037945*a^2 + 1909862879455*a + 5288660409881 (índice j = 5)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 2347678015217*a^2 + 6755730206455*a + 8266046191415
x_i*Δ + m_i = 7271547995351*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5518155607335*a^2
+ 7063282306290*a + 5714827641764 = 2347678015217*a^2 + 6755730206455*a + 8266046191415
Resultado: True
------------------
i=1: q_i = 6533210232240*a^2 + 767044607819*a + 1603836438787
x_i*Δ + m_i = 4925088477414*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3749561930256*a^2
+ 4066543906869*a + 6607613160297 = 6533210232240*a^2 + 767044607819*a + 1603836438787
Resultado: True
------------------
i=2: q_i = 3168708353907*a^2 + 7148450643250*a + 2411310691450
x_i*Δ + m_i = 3253063993110*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1655721241804*a^2
+ 7377509211350*a + 5866668599216 = 3168708353907*a^2 + 7148450643250*a + 2411310691450
Resultado: True
------------------
i=3: q_i = 6612522110013*a^2 + 3598068023231*a + 7080885738025
x_i*Δ + m_i = 8756199863205*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6691801738244*a^2
+ 2355389262838*a + 7737228172449 = 6612522110013*a^2 + 3598068023231*a + 7080885738025
Resultado: True
------------------
i=4: q_i = 7245237959301*a^2 + 4137089653559*a + 3126062460774
x_i*Δ + m_i = 1461394958902*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7834354160932*a^2
+ 6679826349214*a + 8463409740760 = 7245237959301*a^2 + 4137089653559*a + 3126062460774
Resultado: True
------------------
i=5: q_i = 6523270845333*a^2 + 6246891803635*a + 2911872033866
x_i*Δ + m_i = 8244089003708*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7670250484828*a^2
+ 8386029494658*a + 2436803332955 = 6523270845333*a^2 + 6246891803635*a + 2911872033866
Resultado: True
------------------
i=6: q_i = 5243177601416*a^2 + 7470273887903*a + 881947565834
x_i*Δ + m_i = 6377035704846*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4157564534390*a^2
+ 2920693159899*a + 1740643443390 = 5243177601416*a^2 + 7470273887903*a + 881947565834
Resultado: True
------------------
i=7: q_i = 3687416956390*a^2 + 7670084357289*a + 4316231245081
x_i*Δ + m_i = 6821138990438*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 7349064249466*a^2
+ 4208794431338*a + 3184356012681 = 3687416956390*a^2 + 7670084357289*a + 4316231245081
Resultado: True
------------------
i=8: q_i = 4585088465207*a^2 + 770278774453*a + 1576100289780
x_i*Δ + m_i = 5222266180217*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5149986973983*a^2
+ 2371473117297*a + 8743367823258 = 4585088465207*a^2 + 770278774453*a + 1576100289780
Resultado: True
------------------
i=9: q_i = 1758818356565*a^2 + 8470216240639*a + 5559295083039
x_i*Δ + m_i = 2717994838566*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1561680541842*a^2
+ 912981628777*a + 8063712097960 = 1758818356565*a^2 + 8470216240639*a + 5559295083039
Resultado: True
------------------
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
----------------------------FASE MU---------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=6 bem decifrada
------------------
```

i=0: q_i = 3000169461296*a^2 + 7408811610074*a + 5373315287894
x_i*Δ + m_i = 656114884306*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5112987233530*a^2 +
1778987178950*a + 4764069651177 = 3000169461296*a^2 + 7408811610074*a + 5373315287894
Resultado: True
---
i=1: q_i = 7234398696711*a^2 + 8679602766841*a + 466270451637
x_i*Δ + m_i = 5832733368762*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2962232829486*a^2
+ 930945373452*a + 2763548385664 = 7234398696711*a^2 + 8679602766841*a + 466270451637
Resultado: True
---
i=2: q_i = 4116369535418*a^2 + 5468000044567*a + 8419741248940
x_i*Δ + m_i = 6131491830086*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 4518779934899*a^2
+ 1248199233265*a + 15498999765 = 4116369535418*a^2 + 5468000044567*a + 8419741248940
Resultado: True
---
i=3: q_i = 115447808038*a^2 + 1771597527546*a + 5325729171574
x_i*Δ + m_i = 5962965308348*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5114617686436*a^2
+ 7238048010044*a + 2531723757104 = 115447808038*a^2 + 1771597527546*a + 5325729171574
Resultado: True
---
i=4: q_i = 84628142101*a^2 + 2844072975567*a + 6421898287253
x_i*Δ + m_i = 7350134126484*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1096885671420*a^2
+ 6751606182012*a + 818952798383 = 84628142101*a^2 + 2844072975567*a + 6421898287253
Resultado: True
---
i=5: q_i = 2666849780844*a^2 + 2446121157398*a + 7914384398777
x_i*Δ + m_i = 8107810016301*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2363331306858*a^2
+ 3882923034361*a + 1701362320687 = 2666849780844*a^2 + 2446121157398*a + 7914384398777
Resultado: True
---
i=6: q_i = 1569671358744*a^2 + 1347243742653*a + 1388403674270
x_i*Δ + m_i = 5675923787652*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8494747956768*a^2
+ 3519143667397*a + 1557569951394 = 1569671358744*a^2 + 1347243742653*a + 1388403674270
Resultado: True
---
i=7: q_i = 3167416856518*a^2 + 8140225442083*a + 3283949000533
x_i*Δ + m_i = 393705029663*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6939672498281*a^2 +
2074679609875*a + 6470720685311 = 3167416856518*a^2 + 8140225442083*a + 3283949000533
Resultado: True
---
i=8: q_i = 4839237363331*a^2 + 1456446355045*a + 8032809980181
x_i*Δ + m_i = 6640739597080*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2187141187077*a^2
+ 6257129496466*a + 2652443446123 = 4839237363331*a^2 + 1456446355045*a + 8032809980181
Resultado: True
---
i=9: q_i = 6286526559019*a^2 + 7232289377976*a + 3710581720554
x_i*Δ + m_i = 5129261305373*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 8576696810513*a^2
+ 5473130625969*a + 605618253716 = 6286526559019*a^2 + 7232289377976*a + 3710581720554
Resultado: True
---
i=10: q_i = 957908319130*a^2 + 1854577385102*a + 8091805077513
x_i*Δ + m_i = 8759552137374*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5745639278552*a^2
+ 4916078158281*a + 7619593256357 = 957908319130*a^2 + 1854577385102*a + 8091805077513
Resultado: True
---
i=11: q_i = 44720676981*a^2 + 1849248578075*a + 6445282395782
x_i*Δ + m_i = 7563534669842*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2996687332784*a^2
+ 3265976446487*a + 613972718755 = 44720676981*a^2 + 1849248578075*a + 6445282395782
Resultado: True
---
i=12: q_i = 7796313623966*a^2 + 3955363868417*a + 6973347793926
x_i*Δ + m_i = 5774624242457*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 2859395729751*a^2
+ 4933196574000*a + 7400271275318 = 7796313623966*a^2 + 3955363868417*a + 6973347793926
Resultado: True
---
i=13: q_i = 668804726070*a^2 + 3822993289635*a + 8227642550486
x_i*Δ + m_i = 6276538616685*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1754229709834*a^2
+ 3855975436435*a + 5944387515556 = 668804726070*a^2 + 3822993289635*a + 8227642550486
Resultado: True
---
i=14: q_i = 3657238239058*a^2 + 6231144877209*a + 7570394836427
x_i*Δ + m_i = 582053832495*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 3877204011585*a^2 +
3333188155402*a + 6524887566350 = 3657238239058*a^2 + 6231144877209*a + 7570394836427
Resultado: True
---
i=15: q_i = 3844743754930*a^2 + 3266591852598*a + 2847576663593
x_i*Δ + m_i = 4954294700210*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 6848412370386*a^2
+ 7553361527429*a + 7816838771185 = 3844743754930*a^2 + 3266591852598*a + 2847576663593
Resultado: True
---
i=16: q_i = 5891628858677*a^2 + 1516480622987*a + 640218436385
x_i*Δ + m_i = 852171803647*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1524710607928*a^2 +
8447016413158*a + 7260404605704 = 5891628858677*a^2 + 1516480622987*a + 640218436385

```
Resultado: True
------------------
i=17: q_i = 5648242106992*a^2 + 1621889643369*a + 3843946238869
x_i*Δ + m_i = 7088265571148*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 5136644265049*a^2
+ 3711106692145*a + 4360935502731 = 5648242106992*a^2 + 1621889643369*a + 3843946238869
Resultado: True
------------------
i=18: q_i = 269504439131*a^2 + 6074134032786*a + 5306665750124
x_i*Δ + m_i = 4404135109973*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1561738359811*a^2
+ 7016831571149*a + 5802383945579 = 269504439131*a^2 + 6074134032786*a + 5306665750124
Resultado: True
------------------
i=19: q_i = 7261359140322*a^2 + 5705324127653*a + 5772549683073
x_i*Δ + m_i = 4771528196738*3211172037945*a^2 + 1909862879455*a + 5288660409881 + 1995439432524*a^2
+ 6665349764548*a + 59467030045 = 7261359140322*a^2 + 5705324127653*a + 5772549683073
Resultado: True
------------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 1928487385547*a^2 + 1498999662279*a + 8490974930837 == 1928487385547*a^2 + 149899966227
9*a + 8490974930837
Diferença global: 0
j=0: prover_term − verifier_term = 0
```

```
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True

REPETICAO NUMERO 10


--------------------------------------------------------------------------------
----------------------------FASE W----------------------------------------------
--------------------------------------------------------------------------------
------------------
Δ = 351186518504*a^2 + 710356990350*a + 6130922975946 (índice j = 6)
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
------------------
i=0: q_i = 2822214232114*a^2 + 3722986134585*a + 366170757974
x_i*Δ + m_i = 7271547995351*351186518504*a^2 + 710356990350*a + 6130922975946 + 2198737270506*a^2 +
7015019928159*a + 8305153805710 = 2822214232114*a^2 + 3722986134585*a + 366170757974
Resultado: True
------------------
i=1: q_i = 7611560602811*a^2 + 7685092572741*a + 6233322173789
x_i*Δ + m_i = 4925088477414*351186518504*a^2 + 710356990350*a + 6130922975946 + 771332362680*a^2 + 6
648186418244*a + 5465902257469 = 7611560602811*a^2 + 7685092572741*a + 6233322173789
Resultado: True
------------------
i=2: q_i = 2526625544157*a^2 + 2233908937377*a + 8025107882411
x_i*Δ + m_i = 3253063993110*351186518504*a^2 + 710356990350*a + 6130922975946 + 116568383880*a^2 + 8
273594700954*a + 7647648673426 = 2526625544157*a^2 + 2233908937377*a + 8025107882411
Resultado: True
------------------
i=3: q_i = 1810892393846*a^2 + 3327368736805*a + 4538787526826
x_i*Δ + m_i = 8756199863205*351186518504*a^2 + 710356990350*a + 6130922975946 + 6146319257253*a^2 +
2618285114246*a + 2668432263687 = 1810892393846*a^2 + 3327368736805*a + 4538787526826
Resultado: True
------------------
i=4: q_i = 5492744773368*a^2 + 7601041858100*a + 1767631431257
x_i*Δ + m_i = 1461394958902*351186518504*a^2 + 710356990350*a + 6130922975946 + 2172815485210*a^2 +
4955396024583*a + 4676978062127 = 5492744773368*a^2 + 7601041858100*a + 1767631431257
Resultado: True
------------------
i=5: q_i = 5604184548329*a^2 + 6215390590668*a + 2797234860462
x_i*Δ + m_i = 8244089003708*351186518504*a^2 + 710356990350*a + 6130922975946 + 4471409859716*a^2 +
1284996063379*a + 2610501805645 = 5604184548329*a^2 + 6215390590668*a + 2797234860462
Resultado: True
------------------
i=6: q_i = 3748390969026*a^2 + 7171941618209*a + 2121683787855
x_i*Δ + m_i = 6377035704846*351186518504*a^2 + 710356990350*a + 6130922975946 + 3069116267401*a^2 +
7301884563827*a + 5778954040121 = 3748390969026*a^2 + 7171941618209*a + 2121683787855
Resultado: True
------------------
i=7: q_i = 8237800028995*a^2 + 5213075309410*a + 1617656746391
x_i*Δ + m_i = 6821138990438*351186518504*a^2 + 710356990350*a + 6130922975946 + 441648773812*a^2 + 5
074873924092*a + 3275257084611 = 8237800028995*a^2 + 5213075309410*a + 1617656746391
Resultado: True
------------------
i=8: q_i = 1319867304873*a^2 + 4809536335869*a + 8682086339831
x_i*Δ + m_i = 5222266180217*351186518504*a^2 + 710356990350*a + 6130922975946 + 7611118129778*a^2 +
5920460820566*a + 710525663744 = 1319867304873*a^2 + 4809536335869*a + 8682086339831
Resultado: True
------------------
i=9: q_i = 4204738329101*a^2 + 4023271757005*a + 6877041955610
x_i*Δ + m_i = 2717994838566*351186518504*a^2 + 710356990350*a + 6130922975946 + 4019898312050*a^2 +
3913483489039*a + 2002149902634 = 4204738329101*a^2 + 4023271757005*a + 6877041955610
Resultado: True
------------------
```

```
Verificação bem-sucedida? True
--------------------------------------------------------------------------------
----------------------------FASE MU---------------------------------------------
--------------------------------------------------------------------------------
------------------
Verificação OK: todas as somas coincidem com u_i
------------------
Mensagem msg=1 bem decifrada
Mensagem msg=2 bem decifrada
Mensagem msg=3 bem decifrada
Mensagem msg=4 bem decifrada
Mensagem msg=5 bem decifrada
------------------
i=0: q_i = 3595426161425*a^2 + 6762933149292*a + 6636962936958
x_i*Δ + m_i = 7988141654575*351186518504*a^2 + 710356990350*a + 6130922975946 + 5458242228140*a^2 +
6111514062250*a + 7866188106711 = 3595426161425*a^2 + 6762933149292*a + 6636962936958
Resultado: True
------------------
i=1: q_i = 4256408953589*a^2 + 317785312754*a + 2219581469704
x_i*Δ + m_i = 5937688277708*351186518504*a^2 + 710356990350*a + 6130922975946 + 2297460340180*a^2 +
1662993064054*a + 7478279661025 = 4256408953589*a^2 + 317785312754*a + 2219581469704
Resultado: True
------------------
i=2: q_i = 3446110116896*a^2 + 2693237047501*a + 7529963682606
x_i*Δ + m_i = 993363498633*351186518504*a^2 + 710356990350*a + 6130922975946 + 777900581948*a^2 + 56
01710818880*a + 2413799723159 = 3446110116896*a^2 + 2693237047501*a + 7529963682606
Resultado: True
------------------
i=3: q_i = 5129437497401*a^2 + 7657990463184*a + 8399384361068
x_i*Δ + m_i = 6488775360484*351186518504*a^2 + 710356990350*a + 6130922975946 + 4531451669840*a^2 +
747264229120*a + 1351111482197 = 5129437497401*a^2 + 7657990463184*a + 8399384361068
Resultado: True
------------------
i=4: q_i = 161868810317*a^2 + 7739087434767*a + 3544509514127
x_i*Δ + m_i = 8455945340540*351186518504*a^2 + 710356990350*a + 6130922975946 + 875452589940*a^2 + 4
98376832705*a + 1067212580799 = 161868810317*a^2 + 7739087434767*a + 3544509514127
Resultado: True
------------------
i=5: q_i = 3483006565198*a^2 + 5198649933406*a + 6201873297905
x_i*Δ + m_i = 3775026529021*351186518504*a^2 + 710356990350*a + 6130922975946 + 6664718727967*a^2 +
5592651167715*a + 4031779179423 = 3483006565198*a^2 + 5198649933406*a + 6201873297905
Resultado: True
------------------
i=6: q_i = 1349003780070*a^2 + 1434321114278*a + 1126430207130
x_i*Δ + m_i = 948146268287*351186518504*a^2 + 710356990350*a + 6130922975946 + 4621229396854*a^2 + 6
682038124162*a + 2902337058804 = 1349003780070*a^2 + 1434321114278*a + 1126430207130
Resultado: True
------------------
i=7: q_i = 4491401422487*a^2 + 886654390216*a + 8655398342300
x_i*Δ + m_i = 2354514863184*351186518504*a^2 + 710356990350*a + 6130922975946 + 4921699727146*a^2 +
6778326974285*a + 8108122897624 = 4491401422487*a^2 + 886654390216*a + 8655398342300
Resultado: True
------------------
i=8: q_i = 4496042245653*a^2 + 6326258519433*a + 399551440217
x_i*Δ + m_i = 2630405988885*351186518504*a^2 + 710356990350*a + 6130922975946 + 8532818125995*a^2 +
5188231925659*a + 7804463064914 = 4496042245653*a^2 + 6326258519433*a + 399551440217
Resultado: True
------------------
i=9: q_i = 6628002071720*a^2 + 7578583101228*a + 7995802361844
x_i*Δ + m_i = 5746405077358*351186518504*a^2 + 710356990350*a + 6130922975946 + 2653572865831*a^2 +
483440077331*a + 4395639711658 = 6628002071720*a^2 + 7578583101228*a + 7995802361844
Resultado: True
------------------
i=10: q_i = 8368237370296*a^2 + 885655534169*a + 7725942584471
x_i*Δ + m_i = 8087730310953*351186518504*a^2 + 710356990350*a + 6130922975946 + 4326681976993*a^2 +
2821209029501*a + 3987864038670 = 8368237370296*a^2 + 885655534169*a + 7725942584471
Resultado: True
------------------
i=11: q_i = 1537168935616*a^2 + 6555307874951*a + 1928499541786
x_i*Δ + m_i = 1703225367007*351186518504*a^2 + 710356990350*a + 6130922975946 + 5665288491549*a^2 +
8397965826754*a + 5122070727372 = 1537168935616*a^2 + 6555307874951*a + 1928499541786
Resultado: True
------------------
i=12: q_i = 3911583555777*a^2 + 6572624836861*a + 6421100765577
x_i*Δ + m_i = 4843183443094*351186518504*a^2 + 710356990350*a + 6130922975946 + 4001093030534*a^2 +
1061468455194*a + 2331034679627 = 3911583555777*a^2 + 6572624836861*a + 6421100765577
Resultado: True
------------------
i=13: q_i = 3501607799547*a^2 + 2558429667235*a + 1103811587891
x_i*Δ + m_i = 2943635469184*351186518504*a^2 + 710356990350*a + 6130922975946 + 2235838841986*a^2 +
6912647356681*a + 5940237877623 = 3501607799547*a^2 + 2558429667235*a + 1103811587891
Resultado: True
------------------
```

```
i=14: q_i = 1162751564414*a^2 + 5379152506115*a + 5655110134391
x_i*Δ + m_i = 6951410279578*351186518504*a^2 + 710356990350*a + 6130922975946 + 70432988852*a^2 + 84
45592504300*a + 35679319228 = 1162751564414*a^2 + 5379152506115*a + 5655110134391
Resultado: True
------------------
i=15: q_i = 5522342294073*a^2 + 7583086304714*a + 4554301034101
x_i*Δ + m_i = 8049677719739*351186518504*a^2 + 710356990350*a + 6130922975946 + 3923422118883*a^2 +
5531920916216*a + 1158826692821 = 5522342294073*a^2 + 7583086304714*a + 4554301034101
Resultado: True
------------------
i=16: q_i = 7080880404435*a^2 + 1617895250166*a + 2117881936340
x_i*Δ + m_i = 180918877280*351186518504*a^2 + 710356990350*a + 6130922975946 + 6433561727845*a^2 + 7
804572577970*a + 4869336591665 = 7080880404435*a^2 + 1617895250166*a + 2117881936340
Resultado: True
------------------
i=17: q_i = 3326382691231*a^2 + 718647121303*a + 5973124234075
x_i*Δ + m_i = 8682110137008*351186518504*a^2 + 710356990350*a + 6130922975946 + 4615014814571*a^2 +
2741216971897*a + 4986265695357 = 3326382691231*a^2 + 718647121303*a + 5973124234075
Resultado: True
------------------
i=18: q_i = 1282848741505*a^2 + 4034007623347*a + 597560533192
x_i*Δ + m_i = 4078125155625*351186518504*a^2 + 710356990350*a + 6130922975946 + 2878816441985*a^2 +
697428346508*a + 7111368485110 = 1282848741505*a^2 + 4034007623347*a + 597560533192
Resultado: True
------------------
i=19: q_i = 6031009699478*a^2 + 1936681056820*a + 7196387989505
x_i*Δ + m_i = 4106487641302*351186518504*a^2 + 710356990350*a + 6130922975946 + 2687911019642*a^2 +
6794685417282*a + 6610359701711 = 6031009699478*a^2 + 1936681056820*a + 7196387989505
Resultado: True
------------------
Verificação de f_i(1; w):
Para i=0, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=0
Para i=1, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=1
Para i=2, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=2
Para i=3, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=3
Para i=4, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=4
Para i=5, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=5
Para i=6, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=6
Para i=7, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=7
Para i=8, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=8
Para i=9, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=9
Para i=10, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=10
Para i=11, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=11
Para i=12, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=12
Para i=13, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=13
Para i=14, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=14
Para i=15, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=15
Para i=16, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=16
Para i=17, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=17
Para i=18, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=18
Para i=19, f_i(1;w) = 0
OK: f_i(1;w) = 0 para i=19
Verificação de f_i(Δ; ω) == A1_i * Δ + A0_i:
For i=0, check: True
For i=1, check: True
For i=2, check: True
For i=3, check: True
For i=4, check: True
For i=5, check: True
For i=6, check: True
For i=7, check: True
For i=8, check: True
For i=9, check: True
For i=10, check: True
```

```
For i=11, check: True
For i=12, check: True
For i=13, check: True
For i=14, check: True
For i=15, check: True
For i=16, check: True
For i=17, check: True
For i=18, check: True
For i=19, check: True
Verificação: 3141321089487*a^2 + 5898257636650*a + 971296581039 == 3141321089487*a^2 + 5898257636650
*a + 971296581039
Diferença global: 0
j=0: prover_term − verifier_term = 0
j=1: prover_term − verifier_term = 0
j=2: prover_term − verifier_term = 0
j=3: prover_term − verifier_term = 0
j=4: prover_term − verifier_term = 0
j=5: prover_term − verifier_term = 0
j=6: prover_term − verifier_term = 0
j=7: prover_term − verifier_term = 0
j=8: prover_term − verifier_term = 0
j=9: prover_term − verifier_term = 0
j=10: prover_term − verifier_term = 0
j=11: prover_term − verifier_term = 0
j=12: prover_term − verifier_term = 0
j=13: prover_term − verifier_term = 0
j=14: prover_term − verifier_term = 0
j=15: prover_term − verifier_term = 0
j=16: prover_term − verifier_term = 0
j=17: prover_term − verifier_term = 0
j=18: prover_term − verifier_term = 0
j=19: prover_term − verifier_term = 0
Prova válida? True
lista_de_oks =  [True, True, True, True, True, True, True, True, True, True]
```