Теоретические основы HFE криптосистемы

Материал для защиты проекта

1. Что такое HFE (Hidden Field Equations)?

Определение

HFE (Hidden Field Equations) — это криптографическая система с открытым ключом, основанная на многочленных уравнениях над конечными полями. Система была предложена французским криптографом Жаком Патарином в 1996 году.

Основная идея

HFE использует тот факт, что решение многочленных уравнений над конечными полями является сложной вычислительной задачей, в то время как вычисление значения многочлена — простая операция.

Ключевые особенности

- **Асимметричная криптосистема**: разные ключи для шифрования и расшифрования
- **Основана на проблеме решения многочленных уравнений**: сложность решения растет экспоненциально
- **Использует скрытую структуру**: аффинные преобразования скрывают внутреннюю структуру многочлена

2. Конечные поля GF(2^n)

Определение конечного поля

Конечное поле GF(2^n) (поле Галуа) — это поле, содержащее 2^n элементов, где операции сложения и умножения определены специальным образом.

Элементы поля

- Элементы поля можно представить как:
  - Векторы из n бит: $(a_0, a_1, \ldots, a_{n-1})$, где $a_i \in \{0, 1\}$
  - Многочлены степени < n над GF(2)
  - Целые числа от 0 до 2^n - 1

Операции в GF(2^n)

Сложение

- **Определение**: XOR (исключающее ИЛИ)
- **Пример**: $5 + 3 = 5 \oplus 3 = 101_2 \oplus 011_2 = 110_2 = 6$
- **Свойства**:
  - Коммутативность: a + b = b + a
  - Ассоциативность: (a + b) + c = a + (b + c)
  - a + a = 0 (каждый элемент является обратным самому себе)

Умножение

- **Определение**: Умножение многочленов по модулю неприводимого многочлена
- **Неприводимый многочлен**: Многочлен степени n, который нельзя разложить на множители над GF(2)
- **Пример для GF(2^8)**:
  - Неприводимый многочлен: $x^8 + x^4 + x^3 + x + 1$ = 0x11B
  - Умножение выполняется с приведением по этому модулю

Возведение в степень

- Выполняется через быстрое возведение в степень
- Используется для вычисления HFE многочлена

Зачем нужны конечные поля?

- Обеспечивают математическую структуру для криптографических операций
- Позволяют работать с большими числами как с элементами конечного множества
- Операции определены и обратимы (кроме умножения на 0)

3. HFE многочлен

Определение

HFE многочлен — это специальный многочлен над конечным полем GF(2^n) вида:

$$P(x) = \Sigma_{ij}\ a_{ij} \cdot x^{(2^i + 2^j)}$$

где:

- a_{ij} — коэффициенты из GF(2^n)
- i, j — индексы, такие что i ≤ j ≤ d
- d — степень многочлена (ограничена для обеспечения возможности расшифрования)

Упрощенная форма (в нашей реализации)

В упрощенной реализации используется многочлен:

$$P(x) = x^2 + x^4 + x^8 + ... + x^{(2^d)}$$

где степени являются степенями двойки.

Свойства HFE многочлена

1. **Ограниченная степень**: Степень d ограничена, чтобы можно было решить уравнение P(x) = y
2. **Квадратичность**: Многочлен имеет квадратичную структуру
3. **Односторонняя функция**: Легко вычислить P(x), но сложно найти x для заданного y

Вычисление HFE многочлена

```
def _hfe_polynomial(self, x: int) -> int:
    result = 0
    for i in range(1, self.d + 1):
        power = 2 ** i  # Степень: 2, 4, 8, 16, ...
        if power < self.field_size:
            result = field.add(result, field.power(x, power))
    return result
```

Почему HFE многочлен?

- Обеспечивает одностороннюю функцию (легко вычислить, сложно обратить)
- Квадратичная структура позволяет эффективно работать с уравнениями
- Ограниченная степень делает возможным решение уравнений (хотя и сложное)

4. Аффинное преобразование S

Определение

Аффинное преобразование S — это секретное преобразование, применяемое к открытому тексту перед вычислением HFE многочлена.

Математическая форма

```
S: y = S₁ · x + S₀
```

где:

- $\mathbf{S_1}$ — обратимая матрица размера n×n над GF(2)
- $\mathbf{S_0}$ — вектор-столбец размера n над GF(2)
- **x** — входной вектор (открытый текст)
- **y** — выходной вектор

Компоненты преобразования S

Матрица S_1

- **Размер**: n×n бит
- **Свойство**: Обратимая ($\det(S_1) \neq 0$)
- **Генерация**: Случайная обратимая матрица над GF(2)
- **Назначение**: Линейное преобразование входных данных

Вектор S_0

- **Размер**: n бит
- **Генерация**: Случайный вектор
- **Назначение**: Сдвиг (translation) в аффинном преобразовании

Зачем нужно преобразование S?

1. **Скрытие структуры**: Скрывает внутреннюю структуру HFE многочлена
2. **Увеличение сложности**: Делает криптосистему более устойчивой к атакам
3. **Секретный ключ**: Является частью секретного ключа

Реализация

```python
def _affine_transform(self, x: List[int], A: np.ndarray, b: np.ndarray) -> List[int]:
    """Применение аффинного преобразования: y = A*x + b"""
    x_vec = np.array(x, dtype=np.uint8)
    y = (A @ x_vec + b) % 2  # Матричное умножение + сложение по модулю 2
    return y.tolist()
```

Обратное преобразование S^{-1}

Для расшифрования необходимо обратное преобразование:

$$S^{-1}: x = S_1^{-1} \cdot (y - S_0) = S_1^{-1} \cdot y + S_1^{-1} \cdot (-S_0)$$

где S_1^{-1} — обратная матрица к S_1.

5. Аффинное преобразование T

Определение

Аффинное преобразование T — это секретное преобразование, применяемое к результату HFE многочлена для получения шифротекста.

Математическая форма

```
T: y = T₁ · x + T₀
```

где:

- $\mathbf{T_1}$ — обратимая матрица размера n×n над GF(2)
- $\mathbf{T_0}$ — вектор-столбец размера n над GF(2)
- **x** — входной вектор (результат HFE многочлена)
- **y** — выходной вектор (шифротекст)

Компоненты преобразования T

Матрица T_1

- Аналогична матрице S_1
- Обратимая матрица n×n над GF(2)
- Генерируется независимо от S_1

Вектор T_0

- Аналогичен вектору S_0
- Случайный вектор размера n
- Генерируется независимо от S_0

Зачем нужно преобразование T?

1. **Дополнительное скрытие**: Дополнительно скрывает структуру HFE многочлена
2. **Симметрия**: Обеспечивает симметричную структуру (S и T)
3. **Увеличение безопасности**: Два независимых преобразования увеличивают сложность атаки

Обратное преобразование T^{-1}

Для расшифрования:

```
T⁻¹: x = T₁⁻¹ · (y − T₀) = T₁⁻¹ · y + T₁⁻¹ · (−T₀)
```

6. Полный алгоритм шифрования

Пошаговое описание

```
Открытый текст (вектор из n бит)
    ↓
[Шаг 1] Применение S: x' = S₁ · x + S₀
    ↓
[Шаг 2] Преобразование в элемент поля: x' → элемент GF(2^n)
    ↓
[Шаг 3] Вычисление HFE многочлена: y' = P(x')
    ↓
[Шаг 4] Преобразование обратно в вектор: y' → вектор из n бит
    ↓
[Шаг 5] Применение T: y = T₁ · y' + T₀
    ↓
Шифротекст (вектор из n бит)
```

Математическая формула

```
E(x) = T(P(S(x)))
```

где:

- E — функция шифрования
- S — аффинное преобразование S
- P — HFE многочлен
- T — аффинное преобразование T

Пример (концептуальный)

Пусть открытый текст: `x = [1, 0, 1, 0, 1, 0, 1, 0]` (8 бит)

1. **S(x)**: Применяем матрицу S_1 и вектор S_0 → получаем `x'`
2. **P(x')**: Вычисляем HFE многочлен → получаем `y'`
3. **T(y')**: Применяем матрицу T_1 и вектор T_0 → получаем `y` (шифротекст)

7. Алгоритм расшифрования

Пошаговое описание

```
Шифротекст (вектор из n бит)
    ↓
[Шаг 1] Обратное преобразование T: y' = T₁⁻¹ · (y − T₀)
    ↓
[Шаг 2] Преобразование в элемент поля: y' → элемент GF(2^n)
    ↓
[Шаг 3] Решение HFE уравнения: найти x' такой, что P(x') = y'
    ↓
[Шаг 4] Преобразование обратно в вектор: x' → вектор из n бит
    ↓
[Шаг 5] Обратное преобразование S: x = S₁⁻¹ · (x' − S₀)
    ↓
Открытый текст (вектор из n бит)
```

Математическая формула

```
D(y) = S⁻¹(P⁻¹(T⁻¹(y)))
```

где:

- D — функция расшифрования
- T^{-1}, P^{-1}, S^{-1} — обратные преобразования

Решение HFE уравнения

Проблема: Найти x такой, что P(x) = y

Метод в нашей реализации: Перебор всех возможных значений

```python
def _solve_hfe(self, y: int) -> int:
    """Решение уравнения HFE: найти x такой, что P(x) = y"""
    for x in range(self.field_size):  # Перебор всех 2^n значений
        if self._hfe_polynomial(x) == y:
            return x
    return 0  # Если решение не найдено
```

Сложность: O(2^n) в худшем случае

Почему это работает:

- Поле конечно (2^n элементов)
- Гарантированно найдется решение (или его нет)
- Для малых n (например, n=8) перебор выполним

8. Секретные ключи

Состав секретного ключа

Секретный ключ состоит из:

1. **S_1** — обратимая матрица n×n (линейная часть преобразования S)
2. **S_0** — вектор размера n (сдвиг преобразования S)
3. **T_1** — обратимая матрица n×n (линейная часть преобразования T)
4. **T_0** — вектор размера n (сдвиг преобразования T)

Генерация ключей

```
def _generate_keys(self):
    # Генерация S
    self.S1 = self._generate_invertible_matrix()  # Обратимая матрица
    self.S0 = np.random.randint(0, self.field_size, size=self.n)

    # Генерация T
    self.T1 = self._generate_invertible_matrix()  # Обратимая матрица
    self.T0 = np.random.randint(0, self.field_size, size=self.n)

    # Вычисление обратных матриц для расшифрования
    self.S1_inv = self._matrix_inverse(self.S1)
    self.T1_inv = self._matrix_inverse(self.T1)
```

Безопасность ключей

- Ключи должны быть **секретными** (не раскрываются)
- Матрицы должны быть **обратимыми** (иначе расшифрование невозможно)
- Ключи генерируются **случайно** (используется seed для воспроизводимости)

9. Почему HFE безопасна?

Криптографические предположения

1. **Сложность решения многочленных уравнений**: Решение уравнения P(x) = y является сложной задачей
2. **Скрытая структура**: Аффинные преобразования S и T скрывают внутреннюю структуру
3. **Односторонняя функция**: Легко вычислить P(x), но сложно найти x для заданного y

Атаки на HFE

1. **Алгебраические атаки**: Попытка решить систему уравнений
2. **Линейные атаки**: Поиск линейных зависимостей
3. **Дифференциальные атаки**: Анализ различий в шифротекстах

Ограничения нашей реализации

⚠️ **Важно**: Наша реализация является **упрощенной** и **учебной**:

- Используется простой перебор для решения уравнений
- Не оптимизирована для реального использования
- Предназначена для демонстрации параллельных вычислений

10. Математические свойства

Свойства аффинных преобразований

1. **Композиция**: Композиция аффинных преобразований — аффинное преобразование
2. **Обратимость**: Если матрица обратима, преобразование обратимо
3. **Линейность**: Линейная часть сохраняет структуру

Свойства HFE многочлена

1. **Квадратичность**: Многочлен имеет квадратичную структуру
2. **Ограниченная степень**: Степень ограничена для возможности расшифрования
3. **Детерминированность**: Для одного входного значения всегда один результат

Свойства операций в GF(2^n)

1. **Ассоциативность**: $(a + b) + c = a + (b + c)$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
2. **Коммутативность**: $a + b = b + a$, $a \cdot b = b \cdot a$
3. **Дистрибутивность**: $a \cdot (b + c) = a \cdot b + a \cdot c$
4. **Нейтральные элементы**: 0 для сложения, 1 для умножения

11. Визуализация алгоритма

Схема шифрования

```
┌─────────────┐
│ Открытый    │
│ текст       │
│ (n бит)     │
└──────┬──────┘
       │
       ▼
┌─────────────┐
│   S(x)      │  ← Аффинное преобразование S
│  S₁·x + S₀  │
└──────┬──────┘
       │
       ▼
┌─────────────┐
│   P(x')     │  ← HFE многочлен
│  P(x') = y' │
└──────┬──────┘
       │
       ▼
┌─────────────┐
│   T(y')     │  ← Аффинное преобразование T
│  T₁·y' + T₀ │
└──────┬──────┘
       │
       ▼
┌─────────────┐
│ Шифротекст  │
│   (n бит)   │
└─────────────┘
```

Схема расшифрования

```
┌─────────────┐
│ Шифротекст  │
│   (n бит)   │
└─────────────┘
       │
       │
       ▼
┌─────────────┐
│  T⁻¹(y)     │  ← Обратное преобразование T
│ T₁⁻¹·(y−T₀) │
└─────────────┘
       │
       │
       ▼
┌─────────────┐
│  P⁻¹(y')    │  ← Решение HFE уравнения
│  P(x') = y' │  (перебор)
└─────────────┘
       │
       │
       ▼
┌─────────────┐
│  S⁻¹(x')    │  ← Обратное преобразование S
│ S₁⁻¹·(x'−S₀)│
└─────────────┘
       │
       │
       ▼
┌─────────────┐
│ Открытый    │
│ текст       │
│ (n бит)     │
└─────────────┘
```

12. Вычислительная сложность

Шифрование одного блока (n бит)

- **Аффинное преобразование S**: $O(n^2)$ — матричное умножение
- **HFE многочлен**: $O(d)$ — d итераций, где d — степень многочлена
- **Аффинное преобразование T**: $O(n^2)$ — матричное умножение

- **Итого**: $O(n^2 + d)$

Расшифрование одного блока (n бит)

- **Обратное преобразование T**: $O(n^2)$
- **Решение HFE уравнения**: $O(2^n)$ — перебор всех возможных значений
- **Обратное преобразование S**: $O(n^2)$
- **Итого**: $O(2^n)$ — доминирует решение HFE уравнения

Для N байт данных

- **Шифрование**: $O(N \cdot (n^2 + d))$
- **Расшифрование**: $O(N \cdot 2^n)$

Вывод: Расшифрование значительно медленнее шифрования из-за необходимости решения HFE уравнений.

13. Ключевые термины и определения

Конечное поле GF(2^n)

Математическая структура с 2^n элементами, где определены операции сложения и умножения.

Неприводимый многочлен

Многочлен степени n, который нельзя разложить на множители над GF(2). Используется для определения умножения в поле.

Аффинное преобразование

Преобразование вида $y = A \cdot x + b$, где A — матрица, b — вектор. Состоит из линейной части ($A \cdot x$) и сдвига (b).

Обратимая матрица

Матрица, для которой существует обратная матрица. Необходима для возможности расшифрования.

HFE многочлен

Специальный многочлен над конечным полем, используемый в криптосистеме HFE. Имеет ограниченную степень и квадратичную структуру.

Односторонняя функция

Функция, которую легко вычислить, но сложно обратить. HFE многочлен является односторонней функцией.

14. Формулы для запоминания

Аффинное преобразование

$$y = A \cdot x + b$$

Обратное аффинное преобразование

$$x = A^{-1} \cdot (y - b) = A^{-1} \cdot y + A^{-1} \cdot (-b)$$

HFE многочлен (упрощенный)

$$P(x) = \Sigma_i\ x\hat{}(2^i) \quad \text{для } i = 1, 2, ..., d$$

Алгоритм шифрования

$$E(x) = T(P(S(x)))$$

Алгоритм расшифрования

$$D(y) = S^{-1}(P^{-1}(T^{-1}(y)))$$

Сложение в GF(2^n)

```
a + b = a ⊕ b  (XOR)
```

15. Примеры для объяснения

Пример 1: Аффинное преобразование

Дано:

- Матрица S_1 (3×3): `[[1,0,1], [0,1,1], [1,1,0]]`
- Вектор S_0: `[1, 0, 1]`
- Входной вектор x: `[1, 0, 1]`

Вычисление:

```
y = S₁ · x + S₀
y = [[1,0,1], [0,1,1], [1,1,0]] · [1,0,1] + [1,0,1]
y = [1⊕0⊕1, 0⊕0⊕1, 1⊕0⊕0] + [1,0,1]
y = [0, 1, 1] + [1, 0, 1]
y = [0⊕1, 1⊕0, 1⊕1]
y = [1, 1, 0]
```

Пример 2: HFE многочлен

Дано:

- n = 8, d = 3
- x = 5 (в поле $GF(2^8)$)

Вычисление:

```
P(5) = 5² + 5⁴ + 5⁸
     = 25 + 625 + 390625 (в обычной арифметике)
     = вычисление в GF(2⁸) с приведением по модулю
```

Пример 3: Полный цикл шифрования

Открытый текст: `[1, 0, 1, 0, 1, 0, 1, 0]` (байт 0xAA)

1. **S(x)**: `[1,0,1,0,1,0,1,0]` → `[0,1,1,0,0,1,1,0]` (пример)
2. **P(x')**: `[0,1,1,0,0,1,1,0]` → вычисление HFE → `[1,1,0,1,0,1,1,0]`
3. **T(y')**: `[1,1,0,1,0,1,1,0]` → `[0,0,1,1,1,0,0,1]` (шифротекст)

16. Вопросы для самопроверки

1. ✅ Что такое конечное поле GF(2^n)?
2. ✅ Как работает сложение в GF(2^n)?
3. ✅ Что такое неприводимый многочлен?
4. ✅ Что такое аффинное преобразование?
5. ✅ Зачем нужны преобразования S и T?
6. ✅ Как работает HFE многочлен?
7. ✅ Почему расшифрование медленнее шифрования?
8. ✅ Что входит в секретный ключ?
9. ✅ Как решается HFE уравнение?
10. ✅ Какая вычислительная сложность операций?

Удачи на защите! 🎓