1. Криптосистема McEliece

Асимметричная криптосистема, предложенная для анализа, построена на задаче декодирования сигнала, содержащего ошибки.

Основная идея состоит в том, что мы берем код, для которого существует эффективный алгоритм декодирования, и маскируем его под случайный линейный код (умножая на матрицы), и для него уже эффективного алгоритма декодирования не существует, а декодирование произвольного линейного кода является NP-трудной задачей.

В задании для криптосистемы используется уже готовый открытый ключ, который представляет из себя случайную порождающую матрицу G размером $n \times k$, то есть случайный линейный код. Для него не будет существовать эффективного алгоритма декодирования, но для решения задачи нет необходимости расшифровывать сообщения. При этом у криптоаналитика не будет возможности использовать какие-либо особенности кода для взлома всей криптосистемы.

Рассмотрим принцип работы криптосистемы McEliece.

1.1. Генерация параметров криптосистемы

Зафиксируем величины k – битовая размерность открытого текста, n – битовая размерность шифртекста и t – количество ошибочных бит как общие системные параметры.

Сторона A выполняет шаги:

1) Выбирает случайную порождающую матрицу G размером $k \times n$ двоичного линейного $(n; k)$-кода, исправляющего t ошибок, для которого имеется эффективный алгоритм декодирования.

2) Выбирает случайную невырожденную матрицу S размером $k \times k$.

3) Выбирает случайную невырожденную матрицу P размером $n \times n$.

4) Вычисляет $k \times n$ матрицу $\hat{G} = SGP$.

Открытым ключом стороны A будет $(\hat{G}; t)$; закрытым ключом будет $(S; G; P)$.

1.2. Зашифрование

Сторона B выполняет шаги:

1) Получить подлинный открытый ключ A $(\hat{G}; t)$.

2) Представить сообщение двоичной строкой m длины k.

3) Выбрать случайный двоичный вектор z длины n, содержащий ровно t единиц.

4) Вычислить двоичный вектор $c = m\hat{G} + e$.

5) Отправить шифртекст c стороне A.

1.3. Расшифрование

Сторона A выполняет шаги:

1) Вычислить $c' = cP^{-1}$, где P^{-1} – матрица, обратная к P.

2) Использовать алгоритм декодирования для декодирования c' в m'.

3) Вычислить $m = m'S^{-1}$

1.4. Корректность

$$c' = cP^{-1} = (m\hat{G} + e)P^{-1} = (mSGP + e)P^{-1} = (mS)G + eP^{-1};$$

Поскольку $wt(zP^{-1}) \leq t$, ($wt(\cdot)$ – вес вектора (количество единиц)) алгоритм декодирования преобразует c' в $m' = mS$. Оригинальное сообщение m может быть получено из m' умножением на матрицу S^{-1}.

2. Information-set decoding attack

В прикрепленной к задаче статье [1] была рассмотрена атака ISD (Information-set decoding attack, атака по информационным совокупностям). Пусть $\hat{G}$ — открытый ключ криптосистемы McEliece. Тогда для сообщения m зашифрованный текст c вычисляется как $m \cdot \hat{G} + e$. Мы можем записать это выражение в следующем виде:

$$\begin{aligned} m \cdot \hat{G} + e &= m_{1\times k} \cdot (G_1, G_2, ..., G_n)_{k\times n} + (e_1, e_2, ..., e_n)_{1\times n} \\ &= (mG_1, mG_2, ..., mG_n) + (e_1, e_2, ..., e_n) \\ &= (mG_1 + e_1, mG_2 + e_2, ..., mG_n + e_n) \end{aligned}$$

где $1 \leq i \leq n$, G_i представляет i-й столбец сгенерированной матрицы кода, которая задана матрицей публичного ключа.

Важным моментом здесь является то, что вес Хэмминга вектора ошибок $wt(e) = t$ очень мал в сравнении с длинной блока кода. Если криптоаналитик может угадать k из $n - t$ координат по шифртексту c, что соответствует позициям, в которых не было ошибки (в координатах вектора e в этих позициях стоят нули), тогда обозначим через $\overline{c}$ ту часть вектора c, которая содержит только эти позиции (для $\hat{G}$, соответственно, $\overline{\hat{G}}$ – столбцы на тех же позициях):

$$\overline{c} = m \cdot \overline{\hat{G}}$$

Выберем из $\overline{c}$ и из $\overline{\hat{G}}$ k позиций с индексами $\{i_1, i_2, ..., i_k\} \subset \{1, 2, ..., n\}$, чтобы для каждого $1 \leq j \leq k$ выполнялось $e_{i_j} = 0$. Тогда, приведенное выше соотношение для данного набора индексов можно переписать в виде:

$$(c_{i_1}, c_{i_2}, ..., c_{i_k})_{1\times k} = m_{1\times k} \cdot (G_{i_1}, G_{i_2}, ..., G_{i_k})_{k\times k}$$

Это означает, что если матрица $(G_{i_1}, G_{i_2}, ..., G_{i_k})_{k\times k}$ обратима, то сообщение m может быть восстановлено обычным умножением на обратную к ней матрицу.

Мы не знаем заранее, в каких позициях произошли ошибки. Поэтому нам необходимо угадать k позиций из $n-t$ возможных, в которых не произошли ошибки. Таким образом, если мы выбираем столбцы случайно, нам в среднем потребуется $\frac{\binom{n}{k}}{\binom{n-t}{k}}$ предположений, чтобы подобрать подходящую матрицу. Теперь мы можем оценить вычислительную сложность данного шага:

$$k^3 \cdot \frac{\binom{n}{k}}{\binom{n-t}{k}} \approx k^3 \left(1 - \frac{t}{n}\right)^{-k}$$

где k^3 – сложность вычисления обратной матрицы размера $k \times k$.

Вспомним, что открытый ключ $\hat{G}$ – порождающая матрица для некоторого кода, исправляющего t ошибок, а значит минимальное кодовое расстояние больше $2t$. Для двух сообщений u и u' возможно два случая:

Случай 1: Если $u \neq u'$, то при $wt(u\hat{G} + u'\hat{G}) > 2t$. В таком случае, для любого вектора e, для которого $wt(e) = t$, мы получим $wt(u\hat{G} + u'\hat{G} + e) > t$ (оценили по худшему сценарию, когда внесение ошибок e максимально придвинуло сообщения друг к другу в рамках пространства кодовых векторов).

Случай 2: Если $u = u'$, то $u\hat{G} = u'\hat{G}$. В таком случае, если $wt(e) = t$, мы имеем $wt(u\hat{G} + u'\hat{G} + e) = t$ (прибавляем вектор ошибок e к нулевому вектору).

Таким образом, криптоаналитик, получив шифртекст $c = u\hat{G} + e$, должен подобрать сообщение u' и вычислить $wt(u'\hat{G} + c)$. Если вычисленный вес не равен t, то $u \neq u'$. Если вектор ошибок e выбран таким образом, что $wt(e) \leq t$, то для него будет верно утверждение $wt(u\hat{G} + c) \leq t$.

На двух вышеописанных идеях строится атака Lee-Brickell'a.

3. Атака Lee-Brickell'a

Алгоритм Lee-Brickell'a позволяет восстановить вектор ошибок e, используя ISD.
Вход: Матрица $\hat{G}$, шифртекст $c \in \mathbb{F}_q^n$, $p \in \mathbb{N}$, t.
Выход: Вектор ошибок e веса t.

3.1. Алгоритм

1) Собираем случайное подмножество индексов $\{i_1, i_2, ..., i_k\}$ (обозначаем как I – "информационную совокупность") размера k, таким образом, чтобы ранг матрицы $\hat{G}_I = (G_{i_1}, G_{i_2}, ..., G_{i_k})$, $i_j \in I$, был равен k и чтобы она была обратима над полем $GF(2)$, а также составляем соответствующий вектор результирующих бит $c_I = (c_{i_1}, c_{i_2}, ..., c_{i_k})$.

2) Строим матрицу $\hat{G}' = \hat{G}_I^{-1} \cdot \hat{G}$.

3) Рассчитываем $c' = c - c_I \hat{G}'_I$.

4) Если $wt(c') \neq t$, возвращаемся к шагу 1.

4. Related message attack

Применять алгоритм Lee-Brickell'а в чистом виде в данной задаче слишком трудоемко, поэтому мы оптимизируем его, используя для атаки сообщение m, зашифрованное дважды (с разным шумом).

$$\begin{cases} c_1 = m\hat{G} + e_1 \\ c_2 = m\hat{G} + e_2 \end{cases}$$

где $e_1 \neq e_2$.

Пусть $c_j(i)$ обозначает i-ю координату вектора c_j. Тогда составим два множества индексов, L_0 и L_1, следующим образом:

$$L_0 := \{i \in \{1, 2, ..., n\} : c_1(i) + c_2(i) = e_1(i) + e_2(i) = 0\};$$
$$L_1 := \{i \in \{1, 2, ..., n\} : c_1(i) + c_2(i) = e_1(i) + e_2(i) = 1\}.$$

В предположении, что вектора ошибок e_1 и e_2 независимы, вероятность того, что значения битов на i-ой позиции в них совпали, равна:

$$Pr\Big(e_1(i) = 1 = e_2(i)\Big) = \left(\frac{t}{n}\right)^2.$$

Теперь наша цель состоит в том, чтобы оценить вероятность угадывания k неискаженных столбцов из тех, которые проиндексированы множеством L_0. Пусть p_m будет вероятностью того, что ровно m координат e_1 и e_2 будут совпадать и равны 1:

$$p_m = Pr\Big(|\{i : e_1(i) = 1\} \cap \{i : e_2(i) = 1\}| = i\Big) = \frac{\binom{t}{i}\binom{n-k}{t-i}}{\binom{n}{t}}$$

Следовательно, математическое ожидание мощности L_1 равно

$$E(|L_1|) = \sum_{m=0}^{t} (2t - 2m) p_m$$

поскольку каждая новая позиция i, для которой $e_1(i) = 1 = e_2(i)$, сокращает потенциальное количество мест, где могли бы стоять ошибки (мощность L_1), на два.

Модифицированный алгоритм Lee-Brickell'а выглядит следующим образом:

Вход: Матрица $\hat{G}$, шифртекст $c_1, c_2 \in \mathbb{F}_q^n$, $p \in \mathbb{N}$, t.

Выход: Вектор ошибок e веса t.

1) Вычисляем вектор-маску $(c_1(i) == c_2(i))$. По данной маске мы фильтруем столбцы матрицы $\hat{G}$ и элементы вектора c_1, получая матрицу $\widetilde{G}$ и вектор $\widetilde{c}$ соответственно.

2) Собираем случайное подмножество индексов $\{i_1, i_2, ..., i_k\}$ (обозначаем как I) размера k, таким образом, чтобы ранг матрицы $\widetilde{G}_I = (\widetilde{G}_{i_1}, \widetilde{G}_{i_2}, ..., \widetilde{G}_{i_k})$ был равен k, и чтобы она была обратима над $GF(2)$, а также собираем соответствующий вектор результирующих бит $\widetilde{c}_I$.

3) Строим матрицу $\widetilde{G}' = \widetilde{G}_I^{-1} \cdot \hat{G}$.

4) Рассчитываем $c' = c_1 - \widetilde{c}_I \widetilde{G}'_I$.

5) Если $wt(c') \neq t$, возвращаемся к шагу 2.

5. Источник

Code based Cryptography: Classic McEliece