РЕАЛИЗАЦИЯ ЭЛЛИПТИЧЕСКОЙ КРИПТОСИСТЕМЫ С ПАРАЛЛЕЛИЗМОМ МАШИННЫХ ОПЕРАЦИЙ

Н.И. Червяков, М.Г. Бабенко, А.В. Гладков, М.А. Дерябин

Северо-Кавказский федеральный университет, Ставрополь

Рассматриваются проблемы реализации криптографических систем на массивно-параллельных системах. Проводится обзор реализации криптосистем на элиптических кривых. Рассматривается возможность применения системы остаточных классов для повышения эффективности реализации криптографических преобразований.

Введение. Постановка задачи

В современном обществе много внимания уделяется обеспечению безопасности информации. Одним из перспективных способов построения криптосистем с открытым ключом являются криптосистемы, построенные на точках эллиптической кривой. В настоящее время ряд государств, включая Российскую Федерацию, США, Украину и т.д. принимают стандарты по построению цифровых подписей с использованием эллиптических кривых. Также разработаны стандарты шифрования данных на эллиптической кривой IEEE 363 PI (стандартные спецификации для шифрования с открытым ключом), они включают в себя шифрование, цифровую подпись и криптографический ключ механизма согласования с использованием криптосистем на эллиптической кривой (ECC). Вследствие всего вышесказонного становится актуальной задача эффективной реализации арифметики с точками эллиптической кривой с параллелизмом машинных операций.

Основная часть

Модульная арифметика играет центральную роль в реализации ECC. Модульное сложение и умножение по модулю *n* работают как обычные сложение и умножение, за исключением того, что ответ на расчет сводится к его остатку от деления на p. Например, 2 • 4 = 3 (mod 5), потому что 8 имеет остаток 3, когда делится на 5.

Определение 1. Группа точек эллиптической кривой – это точка в бесконечности O и множество решений (x, y) уравнения вида

$$y^2 = x^3 + ax + b(\mathrm{mod}\, p),$$

где a, b – любые два числа из F_p, а p – простое число. Если решение (x, y) удовлетворяет указанному выше уравнению, то (x, y) является точкой на эллиптической кривой.

В аддитивной группе точек эллиптической кривой определена операция сложения $P_1, P_2, P_3 \in EC(F_p)$, где $P_3 = P_1 + P_2$ и координаты точки $P = (x_3, y_3)$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},$$

$$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = \lambda(x_1 - x_3) - y_1.$$

Множество точек на эллиптической кривой образует группу, которая используется в построении эллиптической кривой криптосистемы. ECC могут быть использованы для шифрования и цифровой подписи.

Криптографические ключи генерируются следующим образом

- Выбирается эллиптическая кривая, определенная уравнением $y^2 = x^3 + ax + b \pmod p$. Простое число p должно быть не менее 160 бит.
- Определяется точка P на эллиптической кривой.
- Выбирается целое x в интервале $(1, p)$.
- Вычисляется $Q = xP$.
- Открытый криптографический ключ, являющийся комбинацией эллиптической кривой, выбранной на ней точкой P, простого числа p и Q.
- Закрытый ключ *x*.

Безопасность ECC основана на проблеме решения задачи дискретного логарифмирования эллиптической кривой: возможность определять x при заданных P и Q. Эффективность этой криптосистемы зависит от быстроты вычисления kP для получения некоторого числа x и точки P на эллиптической кривой.

Для создания цифровой подписи сообщения, отправитель должен выполнить следующие действия:

- выбрать целое k в интервале $(1, p-1)$.
- вычислить kP.
- вычислить $k^{-1} \bmod p$.
- вычислить $s = k^{-1}(hex(M) + xr) \bmod p$,

где $hex(M)$ – алгоритм хеширования текста сообщения M, x – закрытый ключ отправителя $r = x_1 \bmod p$.

Цифровой подписью для сообщения m является пара целых чисел (*r*, *s*).

Цифровая подпись отправителя может быть проверена с помощью получателя следующим образом:

- Получить подлинную копию открытого ключа отправителя.
- Убедиться, что r и s – целые числа из интервала $(1, p)$.
- Вычислить $w = s^{-1} \bmod p$.
- Вычислить $hex(M)$.
- Вычислить $u_1 = hex(M)w \bmod p$,

где $hex(M)$ – SHA, вычисляемый по сообщению M.

- Вычислить u_2 = rw mod n.
- Вычислить $u_1 P + u_2 Q = (x_0, y_0) \bmod p$.
- Цифровая подпись отправителя подтверждается, если $r_1 = x_0 \bmod p$.

Трудность решения задачи дискретного логарифмирования эллиптической кривой позволяет достичь более высокого уровня безопасности, при этом имея пропорционально меньший размер ключа шифрования. В свою очередь, чем меньше размер ключа, тем больше вычислительная эффективность. Хотя ECC могут быть реализованы в любом применении криптографии с открытым ключом (например, банкоматы, карточки с балансом телефона, медицинские хранения записей или электронные деньги), использование ECC особенно выгодно в тех случаях, когда возможности хранения, обработки, пропускная способность ограничены. Примерами таких приложений являются беспроводная связь и смарт-карты.

Устройства беспроводной связи и смарт-карты имеют ограниченные возможности хранения, обработки и пропускной способности. Короткий размер ключа ECC снижает как необходимый объем дискового пространства, так и объем данных, которые должны быть переданы. Это также увеличивает эффективность операций с открытым ключом и скорость вычислений. Кроме того, так как эти приложения требуют более высокий уровень безопасности (с более длинными ключами), ECC может продолжать обеспечивать уровень безопасности, эквивалентный другим криптосистемам с открытым ключом, с помощью соразмерно коротких ключей и использования меньшего количества дополнительных системных ресурсов.

В десятичной арифметике логарифмы часто используются для умножения и деления. В СОК используется аналогичный метод, называемый индексными вычислениями [1]. Использование индексного преобразования над полем Галуа $GF(p)$ сводит операции умножения и деления к операциям сложения и вычитания соответственно. Операция умножения является модульной и поэтому может быть реализована в СОК как сложение. С точки зрения аппаратной реализации, сложение в СОК реализуется легче, чем умножение [1, 2]. Деление, напротив, является немодульной операцией в СОК, поэтому использование индексных преобразований над $GF(p)$ значительно улучшит время выполнения операции и снизит аппаратурные затраты.

При построении умножителей, работающих в модулярной арифметике, можно воспользоваться индексным или «дискретно-логарифмическим» представлением операндов и заменить операцию модулярного умножения операцией модулярного сложения индексов или дискретных логарифмов. Индексное представление модулярного числа основывается на понятии первообразного «корня по простому модулю». Таким корнем является целое число, возведение которого в степень 1,2,…, p–1 дает неповторяющиеся вычеты по модулю [1].

Индексный метод позволяет получить очень простую реализацию с помощью метода, аналогичного методу вычисления логарифмов. Умножители по модулю p_i реализуются с помощью табличного поиска при малом p_i (5 бит и менее) и индексного сложения при больших p_i (6 бит и более) [3]. Поэтому в тех случаях, когда диапазон обрабатываемых данных лежит в этих пределах, целесообразно использовать умножители и делители на основе индексного исчисления. Однако индексное счисление может быть использовано только в том случае, когда модуль p_i – простое число.

На практике встречаются также случаи, когда модуль p_i - произвольное число, поэтому появляется необходимость разработать эффективные методы умножения, пригодные при любом модуле, независимо от того, является ли последний простым числом или нет. Так, например, если реализовать умножитель по модулю $p_i = 249$, который не является простым числом, так как $249 = 83 \cdot 3$, то непосредственное использование индексного счисления невозможно. Так как числа 83 и 3 являются простыми, то возможно индексное счисление по разложенным модулям.

При умножении (делении) по модулю p_i сначала определяются индексы чисел, затем эти индексы складываются (вычитаются) по модулю p_i –1 и по обратной таблице определяется окончательный результат [4].

Если модуль не является простым, то необходимо разложить его на меньшие модули, провести вычисления с помощью обычных методов, а затем вновь перейти к исходному модулю. Метод индексного счисления эффективен при обработке данных 6–10 бит, и когда модуль представляет собой простое число [3].

Проблема деления в СОК в общем виде привлекает внимание многих исследователей при разработке высокопроизводительных многомодульных арифметико-логических

устройств (АЛУ). Цифровые системы, построенные на основе арифметики СОК, могут сыграть важную роль в высокоскоростных системах обработки данных в режиме реального времени, с поддержкой параллельной обработки целочисленных данных [5]. Операции сложения, вычитания и умножения, называемые модулярными операциями, могут быть реализованы очень быстро, без распространения переносов [6]. Немодульные операции деления, сравнения чисел, определения знака, определения переполнения диапазона остаются сравнительно медленными [7]. Любое улучшение скорости этих медленных алгоритмов значительно улучшит производительность многомодульных АЛУ.

Некоторые алгоритмы деления в СОК в общем случае были разработаны ранее. Эти алгоритмы могут быть разделены на две категории: с использованием умножения и с использованием вычитания [7]. Большинство алгоритмов на основе умножения предварительно вычисляют обратную величину делителя, после чего эта величина умножается на делимое. Алгоритмы на основе вычитания используют вычитание кратных делителя из делимого до тех пор, пока полученный результат не будет меньше делителя. Некоторые известные алгоритмы деления в СОК на основе умножения изложены в [8]. Все эти алгоритмы используют преобразование в обобщенную позиционную систему счисления (ОПСС) для нахождения обратной величины делителя и сравнения чисел. Кроме того эти алгоритмы являются медленными, так как требуют выполнения большого количества арифметических действий. Некоторые алгоритмы на основе вычитания представлены в [8]. Эти алгоритмы не требуют вычислений в ОПСС, однако используют некоторые другие немодульные операции [4, 5]. Алгоритм с вычитанием, представленный в [9], кажется наиболее привлекательным для применения на практике, поскольку использует эффективный метод проверки четности для сравнения и обнаружения переполнения диапазона. Большинство существующих алгоритмов обладают различными недостатками, что делает их менее пригодными для решения задачи деления в СОК.

Литература

1. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах – М.: Советское радио, 1968. – 440 с.
2. Амербаев В.М., Стемпковский А.Л., Широ Г.Э. Модулярный быстродействующий согласованный фильтр // 50 лет модулярной арифметике. Юбилейная международная научно-техническая конференция. Сборник трудов. – М., 2000. – 775 с.
3. Содерстренд М.А., Верниа К. Недорогой быстродействующий умножитель по модулю и его применение в арифметических устройствах на основе системы счисления в остаточных классах // ТИИЭР, 1980. №4. Т.68.
4. Hung C., Parhami B. Fast RNS division algorithms for fixed divisors with application to RSA encryption // Information Processing Letters 51. 1994. P.163-169.
5. Hung C., Parhami B. An approximate sign detection method for residue numbers and its application to RNS division // Computers Math. Applic. 1994. Vol. 27. №4. P. 23-35.
6. Julien G.A. Residue number scaling and other operations using ROM arrays // IEEE Trans. on Comput. 1978. Vol. 27. №4. P. 325-336.
7. Claudio E.D., Piazza F., Orlandi G. Fast combinatorial RNS processors for DSP applications // IEEE Trans. Comput. 1995. Vol.44. №5. P. 624-633.
8. Червяков Н.И., Лавриненко И.Н., Лавриненко С.В., Мезенцева О.С. Методы и алгоритмы округления, масштабирования и деления чисел в модулярной арифметике

// 50 лет модулярной арифметике. Юбилейная международная научно-техническая конференция. Сборник трудов. М., 2000. С. 291-310.
9. Radhakrishnan D., Yuan Y. Novel approaches to the design of VLSI RNS multipliers // IEEE Trans. Circuits and System. 1992. Vol.39. №1. P.52-57.