Криптография: задачи отображения алгоритмов на вычислительные системы.

Д. А. Желтков, Н. Л. Замарашкин, Е.Е. Тыртышников

ИВМ РАН им. Г.И. Марчука

RSA-232

100988139787192354690956489430946858281823382195557395514112051620583102133852854537436610975715436366491338008491706516992170152473329438927028023438096090980497644054071120196541074755382494867277137407501157718230539834060616 2079

=

296690933320836066036177992424263063474294626252185239440185715741943701947232623907449101125718042744940744527518 91

*

340381617519756343800660949849152142054712176073472317273516341327605070617485265064431443251480888811150838630176 69

Для чего факторизовать RSA числа?

Алгоритм RSA шифрования является одним из базовых криптографических алгоритмов с открытым ключом. Правильное понимание степени его стойкости относится к вопросам национальной безопасности. Стойкость RSA шифрования нарушается, если для базы шифрования n найдено представление

$$n = p \cdot q,$$

с простыми числами p и q.

Таблица рекордов по факторизации RSA чисел

RSA-180	май 2010	И. Поповян (МГУ), А. Тимофеев
RSA-190	ноябрь 2010	И. Поповян (МГУ), А. Тимофеев
RSA-640	ноябрь 2005	F. Bahr, M. Boehm, J. Franke et. all
RSA-200	май 2005	F. Bahr, M. Boehm, J. Franke et. all
RSA-210	сентябрь 2013	R. Propper
RSA-704	июль 2012	S. Bai, E. Thome, P. Zimmermann
RSA-220	май 2016	S. Bai, E. Thome, P. Zimmermann
RSA-230	август 2018	Samuel S. Gross
RSA-768	декабрь 2009	P. Montgomery, A. Lenstra, E. Thome
RSA-232	февраль 2020	Д. Желтков, Н. Замарашкин, С. Матвеев
RSA-240	ноябрь 2019	E. Thome, P. Zimmermann et. all
RSA-250	февраль 2020	E. Thome, P. Zimmermann et. all

ИВМ vs CARAMBA (France + USA + Holland)

к.ф.-м.н. Д.А. Желтков

к.ф.-м.н. С.А. Матвеев

академик Е.Е. Тыртышников

- Суперкопьютер „Жорес“ Сколковского института наук и технологий
- Суперкопьютер „Ломоносов“ суперкомпьютерного центра МГУ им. Ломоносова

ИВМ vs CARAMBA (France + USA + Holland)[1]

Постоянные члены:

1. Xavier Bonnetain – Research scientist, INRIA
2. Sébastien Duval – Assistant professor, Université de Lorraine
3. Pierrick Gaudry – Research director, CNRS
4. Aurore Guillevic – Research scientist, INRIA
5. Virginie Lallemand – Research scientist, CNRS
6. Marine Minier – Professor, Université de Lorraine
7. Cécile Pierrot – Research scientist, INRIA
8. Pierre-Jean Spaenlehauer – Research scientist, INRIA
9. Emmanuel Thome – Research director, INRIA (scientific leader)
10. Paul Zimmermann –Research director, INRIA

Непостоянные члены (аспиранты):

1. Haetham Al Aswad – PhD student, Université de Lorraine
2. Hamid Boukerrou – PhD student, Université de Lorraine
3. Paul Frixons – Postdoctoral fellow, INRIA
4. Antoine Leudière – PhD student, Université de Lorraine
5. Ana Margarita Rodriguez Cordero – PhD student, Université de Lorraine
6. Luc Sanselme – Associate researcher, Lycée Poincaré
7. Quentin Yang – PhD student, Université de Lorraine, INRIA

- CNRS – Национальный центр научных исследований
- INRIA – Национальный институт исследований в информатике и автоматике

[1] **Cryptology, arithmetic: algebraic methods for better algorithms**

Даже при наличии математической записи алгоритма сложное устройство современных вычислительных систем предопределяет задачу поиска эффективного отображения алгоритма на вычислительную систему.

Теоретическая вычислительная математика изобрела большое число способов производить те или иные вычисления. Стандартной мерой эффективности алгоритмов принято считать число операций в них. Сложное устройство современных вычислительных систем существенно меняет эту точку зрения и ставит задачи выбора наиболее эффективного способа вычислений для конкретной вычислительной системы и наиболее эффективной вычислительной системы для данного алгоритма.

Рассмотрим общую структуру метода GNFS (решета общего числового поля):

1. выбор полинома просеивания
2. *просеивание* – получение матрицы соотношений
3. *фильтрация* – редукция матрицы соотношений
4. *линейный этап* – алгоритм решения разреженной линейной системы над полем $\mathbb{F}_2$
5. *алгебраический корень* – алгоритм извлечения квадратного корня в алгебраическом кольце

За построением алгоритма вычислений на каждом этапе метода GNFS стоят сложные алгебраические структуры и глубокие математические результаты. При этом сами конечные вычисления замечательно просты.

1. Идеал $\langle a + b\theta \rangle \in \mathbb{Z}[\theta]$ единственным образом представляется представляется в виде произведения простых идеалов первого порядка
$$\langle a + b\theta \rangle = \prod_s \mathfrak{p}_s, \quad N(\mathfrak{p}_s) = p_s.$$
2. имеется взаимно однозначное соответствие между множеством простых идеалов первого порядка в $\mathbb{Z}[\theta]$ и парами] (r, p), удовлетворяющими соотношению
$$f(r) = 0 \mod p.$$
3. простой идеал первого порядка (r, p) является делителем $\langle a + b\theta \rangle$, если
$$a + br = 0 \mod p.$$

Задаются

1. база $\mathcal{B}_a$ делителей простых идеалов первого порядка в $\mathbb{Z}[\theta]$
2. база $\mathcal{B}_r$ простых делителей в $\mathbb{Z}$

Пара (a, b) взаимно простых чисел называется гладкой, если одновременно

$$a + b\theta = \prod_{(r_s, p_s) \in \mathcal{B}_a} (r_s, p_s)^{\alpha_s},$$

$$a + bm = \prod_{p_t \in \mathcal{B}_r} p_t^{\alpha_t}.$$

Рассматриваем $b \in [0, H]$

Для всех $a_i \in [-H, H]$ взаимно простых с b вычисляются:

1. $N(a_i + b\theta) = (-b)^d f(-\frac{a_i}{b}) \in \mathbb{Z}$
2. $a_i + bm \in \mathbb{Z}$

Таким образом, реальными вычислениями на этапе просеивания являются проверки, что в специальных наборах целых чисел находятся такие, которые делятся на все простые цисла из базы.

Сложность алгоритма определяется размером величины H и, по-общему согласию, оценивается как

$$\exp\left(\left(\left(\frac{64}{9}\right)^{1/3} + \mathcal{O}(1)\right)(\log n)^{1/3}(\log\log n)^{2/3}\right)$$

Размеры баз делителей:

1. база алгебраических делителей $\mathcal{B}_a$ – все простые до 2^{36}
2. база рациональных делителей $\mathcal{B}_r$ – все простые до 2^{35}

Общее число гладких пар, полученное в результате алгоритма просеивания, оказалось равным 5.7 миллиардов. Время, затраченное на поиск гладких пар, ≈ 550 ядро-лет (на суперкомпьютерах „Ломоносов“и „Жорес“). Реальное время просеивания > 1 года.

Этап просеивания идеально соответствует модели неограниченного параллелизма (обмен данными отсутствует)! Скорость работы отдельного узла ограничена пропускной способностью памяти.

Идеальный вычислитель – много слабых узлов с высокой пропускной способностью памяти.

На выходе этапа просеивания для $RSA - 232$ получилась разреженная матрица порядка $5.7 \cdot 10^9$. Применение стандартного метода Гаусса к ней невозможно. Цель фильтрации – получить разреженную матрицу существенно меньших размеров.

Алгоритм фильтрации – структурированное исключение с контролем плотности:

1. удаление дубликатов;
2. удаление синглтонов;
3. удаление клик;
4. выполнение слияний.

Для RSA-232 этап занял 0.1 *ядро-лет на потребовал* 1.5 *TB общей оперативной памяти (исполнен на вычислителе ArCUDA Сколковского технологического университета). Окончательная матрица размера* 317 *миллионов со средним числом ненулей в строке* 170.

Алгоритм фильтрации обладает умеренными параллельными свойствами и предполагает либо наличие достаточношо размера оперативной памяти, либо эффективной out-of-core реализации. Игнорировать требования к вычислительной системе алгоритма фильтрации не получается!

Вычислитель с большим размером общей оперативной памяти.

Матрицы линейных систем, возникающие после применения алгоритмов просеивания и фильтрации, обладают весьма необычными для вычислительной математики свойствами:

1. крайне разреженные (170 элементов в строке длинной 370 миллионов)
2. расположение ненулей в матрице имеет случайный характер
3. системы над конечными полями, где не существует понятия о приближенном решении.

Сложность метода исключений Гаусса для RSA-232 оценивается в $\approx 10^{24}$ *операций. Следовательно, даже для систем с* 10^6 *вычислительных ядер решение займет более* 40 *лет.*

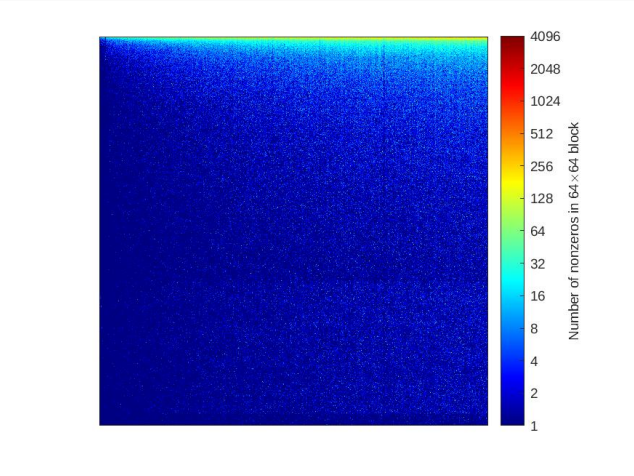


Рис. 1: Портрет матрицы RSA-100, полученный из общего решета числового поля и фильтрации

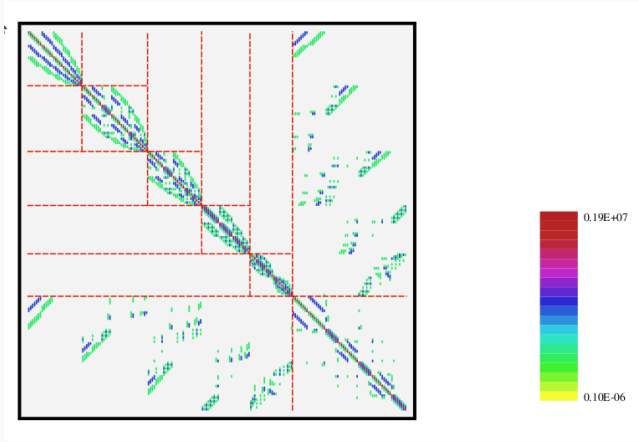


Рис. 2: Портрет конечноразностной аппроксимации дифференциального оператора

Применение стандартных алгоритмов противоречит интуиции вычислителя.

$$\tilde{A}X = 0, \quad \tilde{A} \in \mathbb{Z}_2^{M \times N}, \quad M < N$$

Симметризуем систему

$$A = \tilde{A}^T \tilde{A} \in \mathbb{Z}_2^{N \times N}.$$

Выберем случайный $X_0 \in \mathbb{Z}_2^{N \times 64}$

$$AX_0 = V_0 = W_0 \in \mathbb{Z}_2^{N \times 64}, \text{ и } W_0^T A W_0 \text{ – невырожденная}$$

Далее

$$\begin{aligned} W_i &= V_i S_i, \\ V_{i+1} &= AW_i S_i^T + V_i + W_i C_{i+1,i} + W_{i-1} C_{i+1,i-1} + W_{i-2} C_{i+1,i-2}. \end{aligned}$$

[2] Montgomery, P. L. (1995). A Block Lanczos Algorithm for Finding Dependencies over GF(2). Lecture Notes in Computer Science, 106–120.

Матрицы S_i и $C_{i+1,i}$, $C_{i+1,i-1}$ и $C_{i+1,i-2}$ выбираются из тех условий, что

1. $W_i^T A W_j = 0$, если $i \neq j$
2. $W_i^T A W_i$ являются невырожденными
3. W_i образуют базис в пространстве Крылова $\mathcal{K}(A, V_0)$

Решение системы дается формулой

$$X = \sum_j W_j \left(W_j^T A W_j\right)^{-1} W_i^T W_0,$$

с высокой вероятностью

$$\tilde{A}(X - X_0) = 0$$

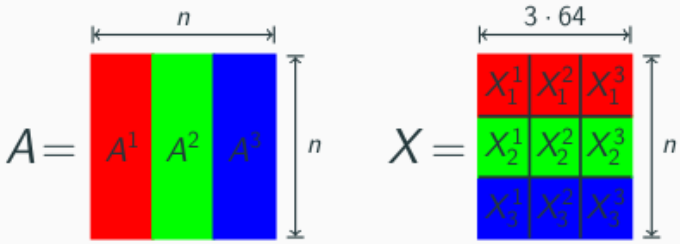
[3] Montgomery, P. L. (1995). A Block Lanczos Algorithm for Finding Dependencies over GF(2). Lecture Notes in Computer Science, 106–120.

$$
\begin{aligned}
W_i &= V_i S_i, \\
V_{i+1} &= A W_i S_i^T + V_i + W_i C_{i+1,i} + W_{i-1} C_{i+1,i-1} + W_{i-2} C_{i+1,i-2}.
\end{aligned}
$$

Математическая запись последовательного алгоритма с гарантированно большим объемом обменов. Число итераций пропорционально размерности системы. Неочивидеый ресурс параллельности.

[4] Montgomery, P. L. (1995). A Block Lanczos Algorithm for Finding Dependencies over GF(2). Lecture Notes in Computer Science, 106–120.

Матрица на блок: разбиение

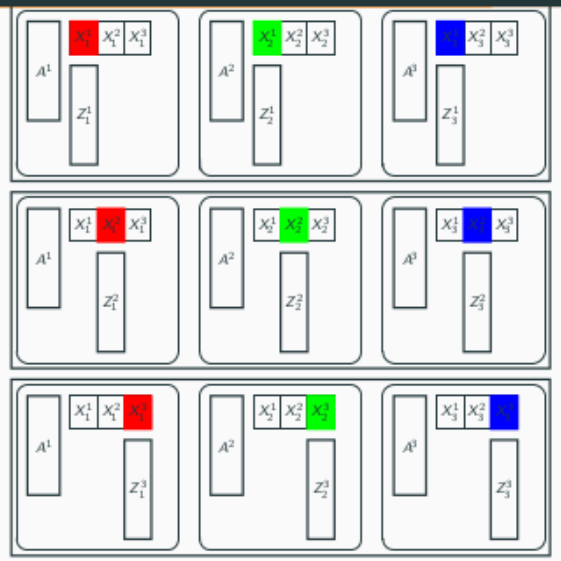


$$\left\{ \begin{array}{c} K_1 = 3, \text{ число блоков в матрице} \\ K_2 = 3, \text{ размер блока} \end{array} \right\} \rightarrow K = K_1 \cdot K_2$$

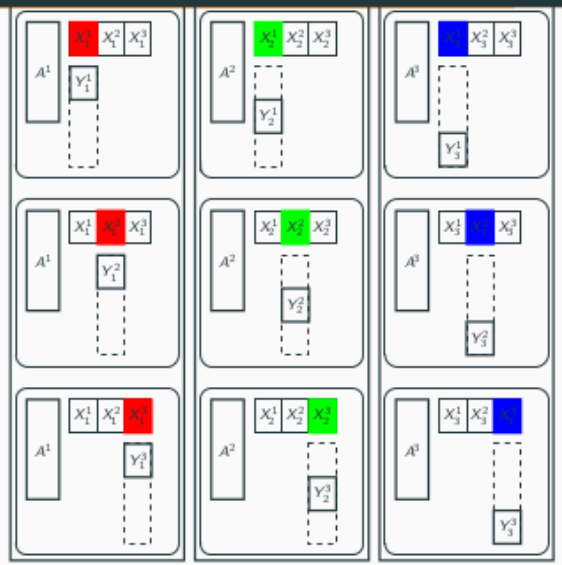
$K = 9$ – общее число вычислительных узлов

33

Матрица на блок: шаг 1

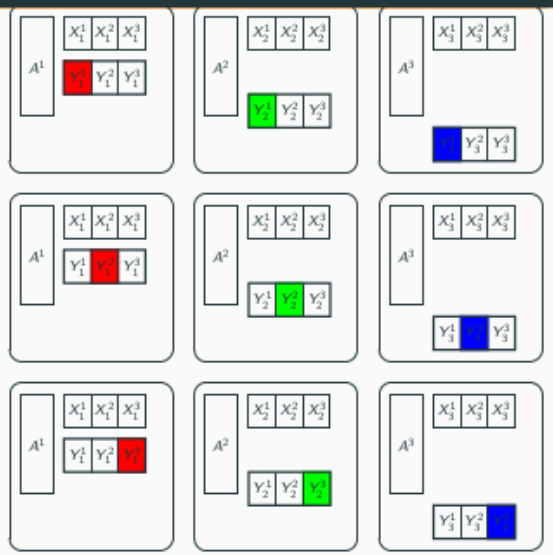


Матрица на блок: шаг 2



35

Матрица на блок: шаг 3



43

Можно показать, что в случае гетерогенных систем перересылки и операции с плотными матрицами и блоками можно замаскировать за умножениями большой разреженной матрицы на блок. Таким образом, построение параллельного алгоритма даже для рекордных размеров неактуально!

	RSA-232	RSA-240
Размер системы	317000000	282000000
Число ненулей в строке	170	200
Размер блока	8	4
Сложность	52 coreyears	69 coreyears

Таблица 1: Сравнение линейного этапа для RSA-232 и RSA-240

Структура поля $\mathbb{F}_2$ позволяет в теории строить быстрые алгоритмы малой сложности для алгебры плотных матриц. Наиболее известным из таких алгоритмов является алгоритм „четырех русских“В. Л. Арлазарова, Е. А. Диница, М. А. Кронрода и И. А. Фараджева.

$n = 2^p$.

$$A = \left[\begin{array}{c|c|c|c} & & & \\ A_1 & A_2 & \dots & A_{n/\log_2(n)} \\ & & & \end{array}\right],$$

$$B = \left[\begin{array}{ccc|c|ccc} & & & B_1 & & & \\ - & - & - & - & - & - & - \\ & & & B_2 & & & \\ - & - & - & - & - & - & - \\ & & & \dots & & & \\ - & - & - & - & - & - & - \\ & & & B_{n/\log_2(n)} & & & \end{array}\right].$$

Используя блочное представление для A и B, получаем

$$AB = \sum_{i=1}^{n/\log_2(n)} A_i B_i.$$

Таблица алгоритма одновременно является и его силой, и его слабостью. Чем больше размер таблицы, тем эффективне метод. С другой стороны, для того, чтобы ускорение было видно на практике, таблица должна лежать быть согласована с иерархией быстрой кэш памяти разных уровней.

Реализация таких алгоритмов затруднена и не имеет высокой степени переносимости.

Даже для матриц размера 2^{16} наилучшее получаемое нами ускорение (результат существенно зависит от типа оборудования) не превосходит [5] значения 2, и, как правило, лежит ниже 1.5.

[5] Д.А. Желтков, З., 2018, 2019

Дополнительные параметры конфигурации для поля GF(2), стр. 1

Объём HBM памяти на процессоре, ГБ	Нет	16	Не важно		Быстрая память нужна только для хранения части вектора (или двух векторов) при вычислении произведения матрицы на вектор
Пропускная способность HBM памяти на процессоре, ГБ / с	Нет	1000	Не важно		Память должна быть значительно быстрее оперативной памяти
Пропускная способность HBM памяти на ускорителях, ГБ / с	Нет		900	2000	Чем больше – тем лучше, наиболее сложная операция зависит от пропускной способности памяти.

11

Параметры конфигурации для поля GF(2), стр. 5

Число уровней кэш-памяти	3		Не важно		Большое число уровней кэш памяти позволяет ускорить наиболее сложную операцию алгоритма – умножение разреженной матрицы на блок, $L1$ и $L2$ кэши позволяют эффективно реализовфвать плотные операции
Объем кэш-памяти L1 на ядро, кБ	32	48	Не важно		Желателен максимально возможный объём
Латентность кэш-памяти L1, тактов	5		Не важно		Важнее пропускная способность

Параметры конфигурации для поля GF(2), стр. 4

Параметр	B1	B2	B3	B4	Комментарии
$V_{ram\ node}$	384	512	256	640	Большой объём оперативной памяти позволяет уменьшить количество узлов, на которые разделена матрица, и, таким образом, уменьшить общий объем пересылок.
$V_{ram\ core}$	6	8	Не важно		
T_{NUMA}	Не важно				Критически важные данные возможно разместить в наиболее быстрой для процессора памяти; для систем с GPU вычисления производятся на GPU

Рассмотрим матрицу $H \in \mathbb{R}^{2\times 2}$ вида

$$H = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Для вектора $\mathrm{vec}(X) \in \{-1, 1\}^{2^n}$ требуется вычислить $\mathrm{vec}(Y) \in \mathbb{R}^{2^n}$ вида

$$\mathrm{vec}(Y) = H^{\otimes n}\mathrm{vec}\ \mathrm{X}.$$

Таким образом, размер задачи $N = 2^n$, измеряемый в **int64** (для задачи с $n = 50$).

1. вычислительная система с общей оперативной памятью $M = 2^m$
2. 2^d дисков $\mathcal{D}_i$, $i = 0, \cdots, 2^d - 1$; чтение независимое (допускается и синхронный и асинхронный режимы);
3. данные на диске хранятся в виде записей (кластеров) B размером 2^b, (то есть чтение осуществляется кластерами);
4. размер дисковой памяти позволяет хранить $2N = 2^{n+1}$ данных

Пусть x_i –i-ая компонента вектора $\mathrm{vec}(X)$. Индекс i полностью задает расположение элемента в модели.

Рассмотрим пример с $b = 3$, $d = 4$, $n = 13$ Тогда элемент x_i с индексом i

$$i = \left[\begin{array}{cccccccccccc} i_0 & i_1 & i_2 & i_3 & i_4 & i_5 & i_6 & i_7 & i_8 & i_9 & i_{10} & i_{11} & i_{12} \end{array} \right],$$

задает положение элемента x_i следующим образом:

$$\begin{array}{cccccc} & & i_0 & i_1 & i_2 & \text{—— позиция элемента в записи (кластере)} \\ & i_3 & i_4 & i_5 & i_6 & \text{—— номер диска} \\ i_7 & i_8 & i_9 & i_{10} & i_{11} & i_{12} \quad \text{—— номер кластера.} \end{array}$$

Положение элемента на дисковом пространстве определяется параметрами системы!

Минимизировать число параллельных копирований с дисковой системы.

Теорема (Vitter, Agrawal) *Нижняя оценка для I/O сложности*

$$I/O\ сложность = \Omega\left(\frac{2^n}{2^b 2^d}\left(\frac{n-b}{m-b}\right)\right)$$

Cormen предложил целую серию алгоритмов аналогичной сложности.

СПАСИБО