Реализация проверки разрешимости Судоку

Федюкович В.Е.
Инфопульс Киев

14 сентября 2019 г.

Аннотация

Предложено решение учебной прикладной задачи о доказательстве существования решения Судоку на основе полиномиального представления множества и библиотеки libsnark, реализующей популярный вариант неинтерактивной системы доказательства.

1 Введение

Вероятностные системы доказательства (interactive proof systems) появились в контексте сложности вычислений, и являются инструментом проверки справедливости утверждений о закрытой информации (защита персональных данных). Доказательства с нулевым разглашением (zero knowledge) появились при анализе нестандартной метрики 'сложности знания' алгоритмов, свойство 'нулевая сложность' требует существование моделирующего алгоритма (simulator). Очень упрощенно, неинтерактивная система доказательства допускает сравнение со схемой электронной подписи и закрытым ключем, имеющим определенную проверяемую структуру. Известен обзор 'Типы нулевого разглашения' [1].

Учебные игровые задачи создают удобные примеры для обсуждения особенностей определений, начиная с 'Нулевое разглашение для Вашего ребенка' [2][QQQ+89]. Задача о проверке разрешимости Судоку была предложена в блоге Fortnow [3], известна также альтернативная проверка на основе совпадения множеств [4][GNPR07]. Реализация была предложена в контексте 'честной' сделки-продажи решения за биткоин [5] как создание и проверка SNARK–доказательства, и стала привлекательной учебной задачей [6].

Термин SNARK стал известным после реализации системы доказательства в виде библиотеки libsnark [7] и создания электронных монет Zcash с публичной проверкой баланса и полномочий без раскрытия получателя,

[1] http://new.math.msu.su/department/dm/dmmc/CONF/11sem_m.htm

[2] http://pages.cs.wisc.edu/~mkowalcz/628.pdf судебное решение о неотличимости
https://medium.com/qed-it/the-incredible-machine-4d1270d7363a

[3] https://blog.computationalcomplexity.org/2006/08/zero-knowledge-sudoku.html

[4] http://www.wisdom.weizmann.ac.il/~naor/PAPERS/SUDOKU_DEMO/ в картинках

[5] https://fc16.ifca.ai/bitcoin/ программа конференции
https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/
https://www.youtube.com/watch?v=ONUsnRgLVB8 демонстрация на Financial Crypto 2016
https://github.com/zcash-hackworks/pay-to-sudoku реализация с libsnark

[6] https://www.youtube.com/watch?v=bD-xTQO_2nAQED-it презентация QED-it

[7] https://github.com/scipr-lab/libsnark реализация и библиография

плательщика и суммы, эмиссия которых оценивается сотнями миллионов долларов. Библиотека стала результатом существенного прогресса в снижении вычислительной сложности проверки утверждений и расширения границ решаемых задач, допускающих сведение к R1CS системе уравнений 'с единственным умножением'. Такая система уравнений допускает сведение к задаче QAP, проверка выполнимости реализуется как делимость полиномов, что допускает вероятностную проверку путем случайного выбора значения аргумента. Проверка SNARK-доказательства реализована с использованием билинейной операции на определенных эллиптических кривых, в том числе семейства MNT [MNT01]. В основном документе приведено подробное пошаговое описание SNARK GROTH16, начиная с генерации общих параметров (Setup); описание рассчитано в первую очередь на математиков, однако изложение достаточно детальное, поэтому будет доступно и новичкам в этой области.

Полиномиальное представление множеств использовалось, в том числе, для решения задачи о вычислении разности множеств (set reconciliation) с низкой коммуникативной сложностью [8][MTZ03]. На основе полиномиального представления множества реализуема схема электронной подписи, допускающая 'небольшие' ошибки в закрытом ключе. Полиномиальное представление графов было предложено в контексте интерактивных систем доказательства для проверки изоморфизма графов и существования цикла Гамильтона [9], и затем использовалось для задачи о допустимой раскраске графа [10][DCF12].

В этой работе предложена проверка разрешимости сценария игры Судоку на основе совпадения множеств (Naor), полиномиального представления множеств, вероятностной проверки утверждений о полиномах, реализации такой проверки как арифметической цепи (circuit) и R1CS-системы; доступен исходный код [11]. Это решение было ранее представлено на Летней школе IFIP 2018.

2 Полиномиальное представление множества

Дня некоторого простого q рассмотрим произвольное конечное множество S различных ненулевых элементов конечного поля $\mathbb{F}_q$. Для каждого такого множества построим следующий характеристический полином над этим полем:

$$f: \quad S \longmapsto f_S(X) = \prod_{a \in S} (1 + Xa) \tag{1}$$

Два таких множества S_1 и S_2 совпадают тогда и только тогда, когда совпадают такие их характеристические полиномы (без доказательства):

$$S_1 = S_2 \iff f_{S_1}(X) \equiv f_{S_2}(X) \tag{2}$$

[8] https://ecommons.cornell.edu/bitstream/handle/1813/5788/2000-1796.pdf

[9] Вопросы оптимизации вычислений, 2007
https://eprint.iacr.org/2008/363
http://conf.fme.vutbr.cz/cecc09/index.php?stranka=lectures

[10] https://dl.acm.org/citation.cfm?id=2368961 MFCS 2012

[11] https://github.com/vadym-f/Sudoku_solvability_proof/

В задаче о $n^2 \times n^2$ Судоку одним из таких множеств является последовательность $\{1 \dots n^2\}$, а вторым - проверяемое множество фактических символов в строках, столбцах или блоках. Ориентированные взвешенные графы также допускают такое полиномиальное представление с многочленами двух переменных.

3 Вероятностная проверка – нулевой многочлен

Проверкой эквивалентности двух полиномов является проверка для разности этих полиномов:

$$f_{S_1}(X) - f_{S_2}(X) \stackrel{?}{\equiv} 0 \tag{3}$$

В некоторых случаях естественная проверка равенства всех коэффициентов нулю является трудно реализуемой, но приемлемым является выбор случайного значения аргумента из некоторого достаточно большого множества:

$$X \to c \in_R C \subseteq \mathbb{F}_q \qquad f_{S_1}(c) - f_{S_2}(c) \stackrel{?}{=} 0 \tag{4}$$

Интерактивная система доказательства Шнорра может рассматриваться как такая проверка утверждения о линейном полиноме – ответе доказывающей стороны, а интерактивные системы о изоморфизме и раскраске графов – как расширение протокола Шнорра с полиномами старших степеней.

Степень полинома как верхняя граница для количества корней дает оценку вероятности ложного правильного ответа (свойство soundness). Обычно на практике характеристикой такого поля является порядок используемой группы, а множество запросов (challenge) проверяющей стороны совпадает с полем, откуда следует ничтожная вероятность такой ошибки, сопоставимая с электронными подписями.

4 R1CS, QAP, и Groth16

Для удобства сопоставления со спецификациями и реализацией libsnark мы пользуемся оригинальными [Gro] обозначениями, за исключением групповой и билинейной операции.

Рассмотрим формализацию прикладной задачи в виде переменных $\{a_i\}$ и R1CS системы уравнений 'с единственным умножением' вида

$$\sum_{i=0}^{m} a_i u_{i,k} \cdot \sum_i a_i v_{i,k} = \sum_i a_i w_{i,k} \tag{5}$$

с дополнительным условием $a_0 = 1$ и условным разделением переменных в духе класса сложности NPна instance ($i = 1 \dots l$) и witness ($i = l+1 \dots m$).

Такая система сводится к QAP–задаче о делимости

$$t(X) \mid \sum_{i=0}^{m} a_i u_i(X) \cdot \sum_i a_i v_i(X) - \sum_i a_i w_i(X) \tag{6}$$

для определенных полиномов $t()$ и $\{u_i(), v_i(), w_i()\}$.

Система доказательства Groth16 состоит их трех алгоритмов: генератор общих параметов (двух публичных ключей), создания и проверки SNARK-доказательства. Входными данными для алгоритма проверки является доказательство, публичная часть задачи (instance) и публичный ключ проверки; для создания – все переменные системы (instance и witness) и публичный ключ создания доказательства. Доказательство Groth16 состоит из двух элементов группы $\mathbb{G}_1$ (точек эллиптической кривой над базовым полем) и еще одного $\mathbb{G}_2$ – над полем расширения.

Открытыми (instance) R1CS переменными в задаче о разрешимости Судоку являются исходный сценарий игры, секретами (witness) – полное решение игры, challenge–значение и дополнительные переменные, необходимые для вычисления значений характеристических полиномов.

5 Проверка разрешимости Судоку

Фотографии как игровой вариант изложения решения на сайте [12] удачно дополняют опубликованный вариант. На каждое игровое поле помещают 3 игральные карты. Для каждой колонки формируется 'вертикальное' множество из n^2 карт, для каждого ряда – 'горизонтальное', для каждого $n \times n$ квадрата – 'блоковое' множество. Проверяется совпадение всех $3n^2$ таких характеристических полиномов с полиномом для множества $\{1 \dots n^2\}$. На рисунке 1 показана арифметическая цепь, иллюстрирующая формализацию вычисления значения 'горизонтального' (row) характеристического полинома $f_r(y, X)$ для строки номер y (то есть, элементов множества $v_{j,y}$) в виде R1CS системы уравнений.

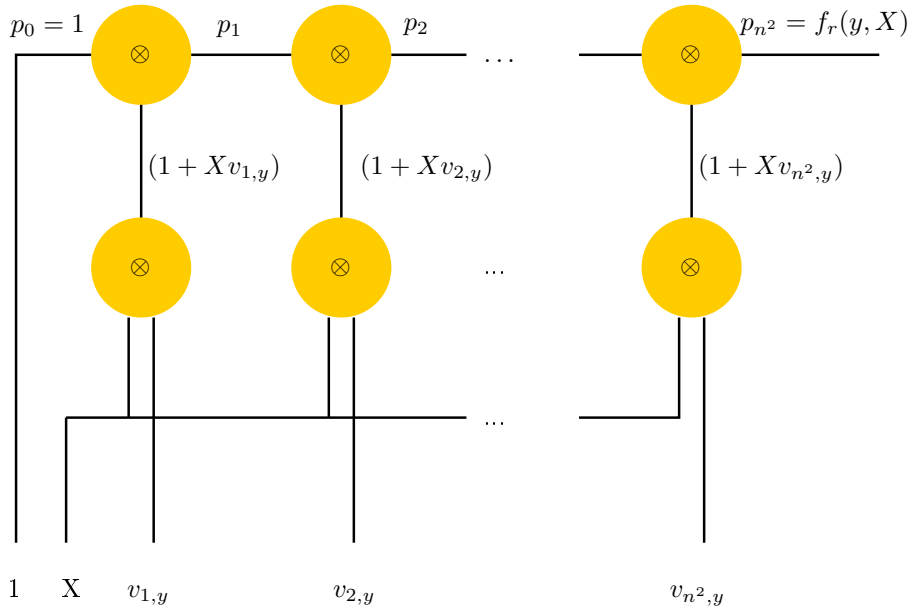

Рис. 1: Арифметическая цепь для характеристического полинома Судоку

[12] http://www.wisdom.weizmann.ac.il/~naor/PAPERS/SUDOKU_DEMO/

6 Использование библиотеки

Библиотека libsnark реализует несколько SNARK–систем, в том числе Groth16. Для вызова генератора публичных ключей `r1cs_gg_ppzksnark_generator<ppT>()` требуется сформировать систему уравнений; уравнения создаются конструкторами `r1cs_constraint<FieldT>()` класса, и добавляются к системе методом `add_r1cs_constraint(,)` класса `protoboard<FieldT>`. Перед вычислением SNARK–доказательства `r1cs_gg_ppzksnark_prover<ppT>()` требуется назначить значения всем R1CS–переменным. Результат да/нет проверки дает `r1cs_gg_ppzksnark_verifier*<ppT>()` (имеется несколько вариантов). Для сложных утверждений используется иерархия C++ классов–гаджетов; имеется несколько готовых классов для утверждений о хэш-функциях, эллиптических кривых и мультипликативных группах.

7 Таблица умножения

В качестве элементарного примера использования библиотеки на рисунке 2 приведен текст C++ программы для получения и проверки SNARK–доказательства. Проверяется существовании некоторого секрета, который при умножении на 4,5,6 дает 12,15,18; используется система доказательства Groth16 и кривая Barreto–Naehrig.

8 Рекурсивная проверка

Уравнение проверки SNARK–доказательства допускает формализацию в виде R1CS–системы, что позволяет строить более сложные утверждения о результате проверки 'предыдущего' SNARK–доказательства. Такая техника позволяет, в том числе, избежать роста истории транзакций с электронными монетами (проект Coda). Реализация требует выбора пары эллиптических кривых [13][BSCTV] с билинейным отображением так, что порядок одной кривой совпадает с характеристикой поля координат другой, и наоборот (например MNT4 и MNT6).

Список литературы

[BSCTV] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. `https://eprint.iacr.org/2014/595`.

[DCF12] Giovanni Di Crescenzo and Vadym Fedyukovych. Zero-knowledge proofs via polynomial representations. In *Proceedings of the 37th International Conference on Mathematical Foundations of Computer Science*, MFCS'12, pages 335–347, 2012.

[GNPR07] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. In *Proceedings of the 4th International*

[13] `https://www.youtube.com/watch?v=cLjfufsi2gM`

Conference on Fun with Algorithms, FUN'07, pages 166–182, Berlin, Heidelberg, 2007. Springer-Verlag.

[Gro] Jens Groth. On the size of pairing–based non–interactive arguments. `https://eprint.iacr.org/2016/260`.

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou TAKANO. New explicit conditions of elliptic curve traces for FR–reduction, 2001. `https://dspace.jaist.ac.jp/dspace/bitstream/10119/4432/1/73-48.pdf`.

[MTZ03] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[QQQ+89] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis C. Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig Guillou, and Thomas A. Berson. How to explain zero-knowledge protocols to your children. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631. Springer, 1989.

```
#include <libff/algebra/curves/alt_bn128/alt_bn128_pp.hpp>
#include <libsnark/gadgetlib1/protoboard.hpp>
 <.../ppzksnark/r1cs_gg_ppzksnark/r1cs_gg_ppzksnark.hpp>
using namespace libsnark;
using namespace std;
#define ppT libff::alt_bn128_pp
#define FieldT libff::Fr<ppT>
int main() {
  libff::alt_bn128_pp::init_public_params();
  protoboard<FieldT> pb;
  pb_variable<FieldT> a, b1, b2, b3, c1, c2, c3;
  b1.allocate(pb, "b1"); b2.allocate(pb, "b2"); b3.allocate(pb, "b3");
  c1.allocate(pb, "c1"); c2.allocate(pb, "c2"); c3.allocate(pb, "c3");
  a.allocate(pb, "a");
  pb.set_input_sizes(6);
  pb.add_r1cs_constraint(r1cs_constraint<FieldT>(a, b1, c1), "eq1");
  pb.add_r1cs_constraint(r1cs_constraint<FieldT>(a, b2, c2), "eq2");
  pb.add_r1cs_constraint(r1cs_constraint<FieldT>(a, b3, c3), "eq3");
  r1cs_gg_ppzksnark_keypair<ppT> kp =
    r1cs_gg_ppzksnark_generator<ppT>(pb.get_constraint_system());
  pb.val(a) = 3;
  pb.val(b1) = 4;  pb.val(b2) = 5;  pb.val(b3) = 6;
  pb.val(c1) = 12; pb.val(c2) = 15; pb.val(c3) = 18;
  if(pb.is_satisfied()) {
    cout << "SAT ok" << endl;
  } else {
    cout << "SAT NOT ok" << endl;
  }
  cout << "Primary input:" << endl << pb.primary_input() << endl;
  cout << "Aux input:" << endl << pb.auxiliary_input() << endl;
  r1cs_gg_ppzksnark_proof<ppT> pi =
    r1cs_gg_ppzksnark_prover<ppT>(kp.pk,
                                  pb.primary_input(),
                                  pb.auxiliary_input());
  bool isvalid =
    r1cs_gg_ppzksnark_verifier_strong_IC<ppT>(kp.vk,
                                              pb.primary_input(),
                                              pi);
  if(isvalid) { cout << "SNARK proof is valid" << endl; }
  pb.val(c2) -= 8;
  if(pb.is_satisfied()) { cout << "modified SAT ok" << endl; }
  bool is_mod_valid =
    r1cs_gg_ppzksnark_verifier_strong_IC<ppT>(kp.vk,
                                              pb.primary_input(),
                                              pi);
  if(is_mod_valid) { cout << "modified instance + SNARK are ok" << endl; }
  return 0;
}
```

Рис. 2: Секретная таблица умножения