Перевірка делегованих обчислень за допомогою додавальної машини

Анатолій Анісімов, Андрій Новокшонов
факультет комп'ютерних наук та кібернетики,
Київський національний університет імені Тараса Шевченка
Київ, Україна
ava@unicyb.kiev.ua, andrey.novokshonov@ukr.net

Verification of delegated computations with addition machines

Anatoliy Anisimov, Andrey Novokshonov
Faculty of Computer Science and Cybernetics,
Taras Shevchenko National University of Kyiv
Kyiv, Ukraine
ava@unicyb.kiev.ua, andrey.novokshonov@ukr.net

***Анотація*— Запропоновано розв'язання проблеми перевірки цілісності арифметичних програм з розгалуженнями і циклами, які виконуються на віддаленому обчислювальному ресурсі. Підхід до розв'язання полягає у заміні арифметичних операцій, таких як множення і ділення, відповідними процедурами додавальної машини (addition machine), введеної Р. Флойдом і Д. Кнутом. Обчислення і послідовність команд підписуються динамічним цифровим підписом, що є гомоморфним за додаванням/відніманням. Для цифрового підпису застосовано модифіковану схему Бенало.**

***Abstract*—A solution of the integrity verification problem for arithmetic programs with branching and looping statements running on a remote computing resource is proposed. The solution is to replace the arithmetic operations such as multiplication and division by corresponding procedures of the addition machine introduced by R. Floyd and D. Knuth. The order of instructions as well as current meanings of variables are signed by dynamic digital signatures, which are homomorphic with respect to addition and subtraction. A modification of the Benaloh scheme is used for digital signatures implementation.**

Ключові слова—верифіковані обчислення; додавальна машина; гомоморфна криптографія; цифровий підпис

Keywords—verifiable computing; addition machine; homomorphic cryptography; digital signature

I.Вступ

Широке поширення та використання хмарних технологій призвело до появи великої кількості сервісів, які дозволяють делегувати дані та обчислення до віддалених обчислювальних ресурсів. Однак оскільки такі ресурси не є довіреними з точки зору користувача (наприклад, через можливі збої в програмному та апаратному забезпеченні, зараження вірусами або навмисні спотворення), то виникає потреба в методах, які могли б забезпечити користувачеві можливість перевірки коректності делегованих обчислень.

Формальне визначення верифікованих обчислень (verifiable computing) вперше було дано Р. Дженнаро, К. Джентрі і Б. Парно в роботі [1], що стала основою для інтенсивних досліджень у цій області, одночасно стимульованих розвитком і масовим впровадженням хмарних технологій такими компаніями як Google, Amazon, Microsoft і іншими.

На сучасному етапі один з основних підходів до вирішення проблем безпеки делегованих обчислень пов'язують із застосуванням гомоморфної криптографії. Гомоморфне шифрування – це окремий вид шифрування, який дозволяє виконувати певні операції над шифротекстом і отримувати зашифрований результат, який під час дешифрування збігається із результатом відповідної операції, виконаної над вихідними даними. Незважаючи на значні стартові досягнення в сфері гомоморфної криптографії, нині відомі методи не мають істотного практичного застосування. Ймовірно, це пов'язано з тим фактом, що властивість повної гомоморфності обчислень щодо основних арифметичних операцій суперечить вимогам криптографічного стійкості і часовим обмеженням обчислювальних процесів.

З іншого боку, наразі відомі та добре вивчені напівгомоморфні криптографічні схеми, в яких гомоморфність виконується тільки за однією з операцій – як правило, за додаванням або множенням. До таких схем відносять багато

відомих систем з публічними ключами: RSA [2] (без недетермінованих добавок), схема Ель-Гамаля [3], схема Пейе [4], схема Бенало [5] та низку інших. Однак реальні обчислення вимагають значно ширшого набору арифметичних операцій, що не дозволяє прямого застосування вищевказаних напівгомоморфних схем.

Таким чином, розглядається проблема цілісності віддалених обчислень, тобто відповідності процесу віддаленого виконання програми користувача її вихідному завданню. Самі дані не шифруються. У моделі обчислень всі змінні та константи – цілі числа; дозволені умовні конструкції типу if-then-else та цикли типу while-do. Дозволені операції – тільки додавання та віднімання на обмеженій кількості регістрів. Така модель обчислень була представлена в роботі Р. Флойда і Д. Кнута [6] та отримала назву «додавальна машина» (addition machine). Незважаючи на обмежений набір операцій додавальної машини, в [6] було показано, що через її операції з прийнятною обчислювальної ефективністю можна виразити більш складні арифметичні операції, такі як множення, цілочисельне ділення, знаходження лишку за модулем, знаходження найбільшого спільного дільника, піднесення до степеня за модулю. Крім того, не створює труднощів розширення області дії додавальної машини на раціональні та дійсні числа за допомогою арифметики з плаваючою комою.

II. Взаємодія користувача з віддаленим обчислювальним ресурсом

Для контролювання цілісності користувач до передавання вихідної програми обчислювальному ресурсу здійснює перетворення програми шляхом вставлення в спеціальні місця зашифрованих значень нових контрольних змінних. Ці контрольні змінні відображають логіку керуючих команд програми і беруть участь у формуванні цифрових підписів. На результати вихідної програми перетворення контрольних змінних не впливає. Віддаленому ресурсу на оброблювання надсилається така видозмінена програма. Отримавши результати обчислень, користувач перевіряє відповідність значень отриманих відкритих значень і розшифрованих значень контрольних змінних. На підставі цього порівняння робиться висновок про коректність виконаних обчислень.

III. Загальний опис методу

Основна ідея запропонованого підходу полягає в заміні основних арифметичних операцій над цілими числами (множення, цілочисельне ділення, обчислення лишку від ділення та інших) відповідними процедурами додавальної машини. При цьому додатково ми використовуємо динамічний цифровий підпис, який здійснюється гомоморфною за додаванням криптографічною функцією на основі функції схеми Бенало [5]. Операції над цифровими підписами виконуються в звичайній модулярній арифметиці обчислювальної машини. Цифровий підпис змінної повинен враховувати три ідентифікуючі показники: поточне значення змінної, ім'я змінної та місце в програмі, де ця змінна змінювала своє значення. Дані вихідної базової програми не шифруються.

Нехай f – недетермінований відображення множини цілих чисел Z до мультиплікативної групи лишків за модулем n, $f: Z \to Z_n^*$. Відображення f побудовано на основі односторонньої функції з потайним входом (trapdoor function). Це означає, що значення $y = f(x)$ легко обчислюється, але ефективне знаходження x за y вимагає знання секретних параметрів (лазівки), відомих тільки користувачу.

Припускаємо, що f задає відображення адитивної групи цілих чисел Z до мультиплікативної групи лишків Z_n^*, яке гомоморфне за додаванням, тобто

$$f(x+y) = f(x) f(y), \quad f(x-y) = f(x) f^{-1}(y).$$

Тут і далі під множенням значень функції f розуміємо операцію множення в групі Z_n^*. На Z_n^* визначено верифікаційне відображення $V: Z_n^* \to Z_n^*$. Вважаємо, що пара $(x,y), x \in Z, y \in Z_n^*$, коректна, якщо $y = f(x)$.

Вибрана наступна схема відображення f. Для $x \in Z$ визначаємо $f(x) = g^{ax} s \bmod n$, де s – довільний елемент власної підгрупи S групи Z_n^*, g – фіксований елемент, $g \notin S$, a – фіксований секретний параметр. Верифікаційна функція V задається формулою $V(y) = y^c \bmod n$, де c – порядок підгрупи S. Параметри g, a, c, S – секретні, відомі тільки користувачу. Верифікація значення змінної x виконується перевіркою співвідношення $E^c(x) = k_x f^c(x) \bmod n$, де $E(x)$ – цифровий підпис x, який обчислюється в процесі виконання програми, k_x – константа, що ідентифікує ім'я змінної x. Стартовим цифровим підписом значення x вважаємо значення $f(x)$. У подальшому до цього підпису буде додана ідентифікація імені змінної.

Вважаємо, що, знаючи значення змінної x, що приймає значення з Z, і значення $y = f(x)$ з Z_n^*, зловмисник не може здійснити підміну пари цих значень, оскільки він не має доступу до закритих параметрів f. Відображення f може розглядатися як динамічний варіант хеш-функції.

Для захисту від спотворення найменувань змінних додатково запроваджується ідентифікуючий цифровий підпис $Id(x)$ імені змінної x, $Id(x) = f(r_x)$, де $r_x \in Z$ – випадково вибраний секретний параметр, r_x – константа, що є фіксованою для кожної змінної x.

Цифровий підпис змінної x визначається виразом:

$$E(x) = Id(x)\, f(x) = f(r_x)\, f(x),$$
$$E(-x) = E^{-1}(x) = f^{-1}(r_x)\, f^{-1}(x).$$

Зміна цифрового підпису $E(z)$ після виконання команди присвоювання додавальної машини визначається наступним чином:

а) команда: $z \leftarrow x + y$;

підпис:

$$E(z) = Id(z)\, Id^{-1}(x)\, Id^{-1}(y)\, E(x)\, E(y). \quad (1)$$

б) команда: $z \leftarrow x - y$;

підпис:

$$E(z) = Id(z)\, Id^{-1}(x)\, Id(y)\, E(x)\, E^{-1}(y). \quad (2)$$

в) команда: $z \leftarrow x$;

підпис:

$$E(z) = Id(z)\, Id^{-1}(x)\, E(x). \quad (3)$$

Виконання кожної команди присвоювання додавальної машини супроводжується попереднім обчисленням відповідного цифрового підпису (1), (2) або (3), асоційованого з цією командою. При цьому результуюча змінна z зв'язується своїм ідентифікатором і цифровим підписом отриманого числового значення. Початкове значення x підписується $E(x)$.

Для контролю цілісності команд та порядку їхнього слідування при виконанні програми здійснюється попередня числова розмітка всіх команд програми. Кожній команді зіставляється унікальне ціле додатне число. Вибір нумеруючих чисел вважається випадковим. Команді з вибраним номером i зіставляється константний ідентифікатор $Id(i) = f(i)$.

Результуючій змінній z, що бере участь у команді присвоювання з номером i, відповідає цифрове значення $Id^{-1}(i)E(z)$. Таким чином, нумерованій команді $i: z \leftarrow x + y$ відповідає значення

$$Id^{-1}(i)\, Id(z)\, Id^{-1}(x)\, Id^{-1}(y)\, E(x)\, E(y).$$

Для перевірки правильності переходів від команди до наступної команди вводиться нова змінна H, яка описує історію виконання команд. Початкове значення H дорівнює $f(i_0)\, f(i_1)$, де i_0 і i_1 – випадково вибрані секретні числа, i_1 – ідентифікаційний номер першої виконаної команди, i_0 – стартове число, що приписується початку програми.

Якщо після виконання j-ої команди відбувається перехід до i-ої команди, то виконується присвоювання $H \leftarrow Id^{-1}(j)\, Id(i)\, H$. Така команда присвоювання вставляється безпосередньо після j-ої команди вихідної програми. Звідси випливає, що поточне значення H у момент виконання i-ої команди дорівнює $f(i_0)\, Id(i) = f(i_0)\, f(i)$.

Таким чином, якщо у вихідній програмі i-а команда має вигляд $z \leftarrow x + y$, то у модифікованій програмі такому фрагменту відповідає блок:

$$H \leftarrow Id^{-1}(j)\, Id(i)\, H;$$
$$E(z) \leftarrow H f^{-1}(i_0)\, Id^{-1}(i)\, Id(z)\, Id^{-1}(x)\, Id^{-1} E(x)\, E(y); \quad (4)$$
$$z \leftarrow x + y.$$

Тут j – номер команди, що передує команді з номером i. Не важко перевірити, що в (4) вираз у правій частини оператора присвоювання $E(z)$ дорівнює $Id(z)\, f(x)\, f(y) = E(z)$.

Значення констант $c_{ji} = Id^{-1}(j)\, Id(i)$ і $c_i = f^{-1}(i_0) Id^{-1}(i) Id(z) Id^{-1}(x) Id^{-1}(y)$ обчислюються заздалегідь, після виконання розмітки. Зауважимо, що коефіцієнт c_i не залежить від попереднього стану історії. Зі стану j в залежності від умов можна перейти до різних станів. Ці стани однозначно визначаються номером j і відповідними умовами.

Після виконання всіх необхідних вставок контролюючих змінних і відповідних команд присвоювання введена розмітка знищується.

Для контроля правильності виконання умовних переходів вводяться додаткові змінні, у яких зберігаються значення змінних після виконання останніх входів/виходів до умови. Зберігаються значення тільки останніх входів/виходів до умови. Вважається, що якщо була вставлена нова невідповідна умова, то вихідна умова хоча б при одному з останніх виконувань буде спотворена. Це буде виявлено на етапі перевірки після отримання результатів виконання програми.

IV. Висновки

Запропоновано підхід до розв'язання перевірки цілісності віддалених арифметичних обчислень над цілочисельними даними, які використовують умовні розгалуження та цикли. Підхід полягає в заміні основних арифметичних операцій (додавання/віднімання, множення/ділення) відповідними процедурами додавальної машини Флойда-Кнута та використанні динамічного цифрового підпису, що виконується гомоморфною за додаванням криптографічною функцією схеми Бенало. Розроблено верифікаційні умови, які дозволяють контролювати правильність виконання умовних розгалужень і операторів циклу.

Література References

[1] R. Gennaro, C. Gentry, B. Parno, "Non-interactive verifiable computing: outsourcing computation to untrusted workers," *Advances in Cryptology. CRYPTO 2010*, pp. 465–482, 2010.

[2] R. L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, issue 2, pp. 120-126, 1978.

[3] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, issue 4, pp. 469-472, 1985.

[4] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg*, pp. 223-238, 1999.

[5] J. Benaloh, "Dense probabilistic encryption," *Proceedings of the workshop on selected areas of cryptography*, pp. 120-128, 1994.

[6] R. W. Floyd, D. E. Knuth, "Addition machines," *SIAM Journal on Computing*, vol. 19, issue 2, pp. 329-340, 1990