Асиметричні криптосистеми та протоколи
Конспект лекцій

Сергій Яковлєв та група ФІ-33

21 січня 2017 р.

Зміст

Передмова

Лекції жодним чином не редагувались (окрім правок, необхідних для компіляції) та наразі надаються as is.

Розділ 1

Важкооборотні функції та важкооборотні функції із секретом

1.1 Односторонні функції. Схема розподілення ключів відкритими каналами.

(Автор: Катя Астаф'єва. Помітно редагувалось Грубіяном Євгеном.)
(Версія від 19 січня 2017 р.)

У минулому семестрі ми познайомилися із симетричною криптографією. В цьому семестрі ми вивчатимемо порівняно молодий розділ криптографії - *асиметричні криптосистеми та протоколи*. Нагадаємо що симетрична криптографія передбачає наявність єдиного ключа для зашифрування і розшифрування даних, тобто сторони, які хотіли би вести захищену комунікацію мають один і той самий ключ.

В 60-70х роках XX ст. виникло 2 основні проблеми, з якими симетричній криптографії було все важче і важче справлятися. Це спрвокувало появу нової гілки в криптографії - *асиметричної криптографії*.

Розлянемо детальніше ці проблеми:

– Розповсюдження(передача) таємного ключа.

  У симетричній криптографії відправник і отримувач повинні мати один і той самий таємний ключ (який знають лише вони і ніхто інший), котрий передається по закритому каналу(часто – перевозили вручну). Проте, раніше криптографією займалися лише військово-дипломатичні відомства. На сьогодні, захистом інформації займаються не лише вони. (Буквально будь-яка галузь науки, промисловості, усі сфери). Тому, експоненційно росте кількість користувачів. Розглянемо простий приклад.

  Приклад 1.1. 10 тис. користувачів хочуть спілкуватися незалежно. Скільки їм потрібно ключів?

$$C_{10000}^2 = \frac{10^4(10^4-1)}{2} = 5(10^7) = 50000000$$

  Висновок: Кількість симетричних ключів росте експоненційно, тому виникає питання яким чином вони будуть доставлятись всім користувачам абсолютно секретно.

– Друга проблема, що важко розв'язувалася симетричною криптографією – автентифікація користувачів. Автентифікація – підтвердження достовірності автора повідомлення. Маючи сектретний ключ, будь-яка сторона може претендувати на автентичність своїх

криптограм, проте перевірити це зможуть тільки інші власники цього ж секретного ключа, розшифрувавши криптограму і ніхто інший.

В асиметричній криптографії основне поняття, на якому базується стійкість її протоколів та алгоритмів – *одностороння функція.*

У 1976 році з'являється стаття Діффі і Хелмана «Новий напрямок у криптографії». Стаття повністю виправдала назву. У ній введено поняття односторонньої функції, запропонована конкретна одностороння функція та схема розподілення ключів відкритими каналами, тобто схема криптографії з відкритими ключами. Таким чином вони розв'язали першу проблему.

Виявляється, таємні ключі можна не передавати закритими каналами. Це виглядало абсурдно. Але як ми побачимо згодом – таємні ключі не розповсюджуються відкритими каналами, а передається певна інформація, що дозволяє ці ключі побудувати так, що не дивлячись на передачу відкритими каналами вони залишаються у таємниці. Для того щоб вирішити таке питання Діффі і Хелман запропонували нове поняття - «одностороння функція» та привели приклад функції, що можливо є односторонньою. Розглянемо означення, що було ними запропоноване.

Визначення 1.1. *Односторонньою функцією* називається відображення на скінченних множинах $f(x) : X \to Y$, таке, що :

1. $\forall x \in X$: існує поліноміальний алгоритм обчислення $y = f(x)$
2. Для майже всіх $y \in Y$: не існує поліноміального (ефективного) алгоритму обчислення оберненої функції $f^{-1}(y)$.

Зауважимо, що X, Y з практичних міркувань стійкості досить потужні множини.

Взагалі кажучи, $f(x)$ – не обовязково бієкція.

Поняття оберненої функції ми розуміємо в узагальненому сенсі(тобто, всі прообрази). Коли говорять, що неможливо ефективно обчислити обернену функцію – мають на увазі, що неможливо ефективно обчислити навіть 1 із таких прообразів.

Чому не для всіх y, а майже для всіх? Якщо казати про практику застосування, то функції завжди мають слабкі(нерухомі) точки. Тобто, ми шифруємо, а текст лишається таким самим. Але, це не головне у слові «майже». В одну сторону функція обчислюється дуже швидко, у іншу – ні. Якщо так – побудуємо табличку. Візьмемо x і обчислимо y. Якщо хтось захоче обернути функцію для конкретного y – він подивиться відповідний x у таблиці.

Видно, що потужність повинна бути досить великою, задля того, щоб таблиця, яку можливо побудувати містила у собі мізерну часту усіх можливих точок.

Визначення 1.2. *Функція Діффі-Хелмана дискретного піднесення до степеня* має вигляд:

$$y = \alpha^x \bmod p\,,$$

де p – деяке просте число, а α – примітивний елемент поля $\mathbb{F}_p$

Область визначення: $X = \{1,2,...,p-1\}$
Область значень: $Y = \{1,2,...,p-1\}$, тобто X = Y
На множині X функція (1) – бієкція, тобто взаємно-обнозначне відображення, в силу примітивності α

На практиці число p має доволі велику довжину, порядка 1024 чи 2048 двійкових розрядів. Існують алгоритми, наприклад Міллера- Рабіна, Соловея - Штрасена та ін. для пошуку таких великих простих чисел за досить ефективний час.

Щодо вибору примітивного елементу використаємо той факт, що α - примітивний тоді і тільки тоді коли $\alpha^{\frac{p-1}{p_i}} \not\equiv 1 \bmod p, i = \overline{1,\,r}$, де $p-1 = \prod_{i=1}^{r} p_i^{k_i}$, p_i - прості, не рівні.

Зауваження. На сьогоднішній день не доведено існування бодай одної односторонньої функції. Ця проблема пов'язана з набагато глибшою проблемаю $P \neq NP$, що поки не розв'язана. Тому приведена вище функція називається не односторонньою, а *кандидатом в односторонні.*

Оцінимо складність обчислення функції Діффі-Хелмана і оберненої до неї.

– **Складність обчислення прямої функції**
$\forall x \in X$ запишемо x = $\sum_{i=0}^{r-1} x_i 2^i$, r - число двійкових розрядів, тоді

$$f(x) = \alpha^{\sum_{i=0}^{r-1} x_i 2^i} = \alpha^{x_0}(\alpha^2)^{x_1} \dots (\alpha^{2^{r-1}})^{x_{r-1}} \bmod p$$

Складність обчислення в такому випадку оцінюється зверху:

$$L_1 \leq 2(r-1) \leq 2[log\, p] \approx 2\log_2 p$$

– **Складність обернення функції Діффі-Хелмана(дискертне логарифмування)**
Даємо деякі оцінки складності алгоритмів дискретного логарифмування без доведень:

$L_2 = O(\sqrt{p})$ - не найшвидший, але можливий для реалізації алгоритм дискретного логарифмування.

$L_3 = exp\{(c_0 + o(1))ln^{1/3}p(lnlnp)^{2/3}\}$, де $c_0 \approx 1{,}923$ - вважають найшвидшим.

Приклад 1.2. Оцінка складності для числа порядку 1024 біт:
$L_1 \leq 2log10^{300} \approx 600 \cdot 3{,}3 \approx 2000$ - швидко
$L_2 = O(\sqrt{10^{300}}) = O(10^{150})$ - число операцій дуже велике.
$L_3 \approx 10^{30}$ - значно менше, але все ще недосяжне для сучасної техніки.

Нехай 2 користувача: **A** і **B** вирішили побудувати секретний ключ, використовуючи відкритий канал.

Алгоритм 1.1 (Схема Діффі-Хелмана розподілу ключів по відкритим каналам)**.**

1. **A** і **B** обирають, просте p і примітивний елемент α.
2. **A** генерує випадкове $x_A \in \{2{,}3, \dots, p-2\}$, x_A – секрет **A**.
   Обчислює $\alpha^{x_A} \bmod p = y_A$ та передає y_A до **B** по відкритому каналу.
3. **B** генерує випадкове $x_B \in \{2{,}3, \dots, p-2\}$, x_B – секрет **B**.
   Обчислює $\alpha^{x_B} \bmod p = y_B$ та передає y_B до **A** по відкритому каналу.
4. **A** обчислює $y_B^{x_A} \bmod p = z_A = k$
5. **B** обчислює $y_A^{x_B} \bmod p = z_B = k$

k - спільний секрет.

Перевіримо що $z_A = z_B$

$$z_A = y_B^{x_A} \bmod p = (\alpha^{x_B})^{x_A} \bmod p = \alpha^{x_A x_B} \bmod p = k$$

$$z_B = y_A^{x_B} \bmod p = (\alpha^{x_A})^{x_B} \bmod p = \alpha^{x_A x_B} \bmod p = k$$

Криптоаналітик **E** знає: p, α, y_A, y_B(перехоплені). Чи зможе він обчислити k?

Варіант обчислення k: із $\alpha^{x_A} mod p = y_A$ знаходимо x_A і обчислюємо $y_B^{x_A} mod p = k$. Але спроби криптоаналітика знайти x_A будуть марними, оскільки це еквівалентно знаходженню дискретного логарифму, що є складною задачею.

Спільний секрет k має довжину 1024 біта з імовірністю трохи більшою за 50%, 1023 з імовірністю трохи меншою ніж 50%. Таким чином можна не використовувати закритий канал.

Покажемо слабкість даного алгоритму до атак типу людина посередині(*Man in the middle attack*).

Алгоритм 1.2 (Атака на протокол Діффі-Хелмана типу людина посередині)**.**

1. **A** обчислює $\alpha^{x_A} \bmod p = y_A$, надсилає **B**
2. **B** обчислює $\alpha^{x_B} \bmod p = y_B$, надсилає **A**
3. Криптоаналітик **E**, котрий перехоплює y_A, y_B обирає своє x_E , обчислює $\alpha^{x_E} mod p = y_E$ та відправляє його **A** і **B**.

   У результаті **A** і **E** побудували спільний ключ, **B** і **E** також побудували спільний ключ. **A** і **B** починають комунікувати, але все перехоплює **E**, розшифровує, можливо модифікує, перешифровує на спільних ключах і надсилає другому абонентові, при цьому присутність криптоаналітика ніяк не помітна для абонентів **A** та **B**.

Таким чином схема Діффі-Хелмана не вирішує задачі автентифікації.

1.2 Схема шифрування Мессі-Омури

(Автор: Всеволод Бахтігозін. Редагувалось.)
(Версія від 20 січня 2017 р.)

Задача: A потрібно відправити ШТ по відкритому каналу B. A і B попередньо вибирають p – велике, просте, відкрите. Нехай ВТ M задовольняє умові: $1 < M < p$.

Алгоритм 1.3 (Мессі-Омури)**.**

1. A: Вибирає довільне $x : (x,p-1) = 1$, обчислює $M^x \bmod p = z_1$ і відсилає z_1 до B.
2. B: Вибирає довільне $y : (y,p-1) = 1$, обчислює $z_1^y \bmod p = z_2$ і відсилає z_2 до A.
3. A: Обчислює $x^{-1} \bmod (p-1), z_2^{x^{-1}} \bmod p = z_3$. Відсилає z_3 до B.
4. B: Обчислює $y^{-1} \bmod (p-1), z_3^{y^{-1}} \bmod p = z_4$.

Покажемо, що $M = z_4$.

$$
\begin{aligned}
z_4 &= z_3^{y^{-1}} \bmod p = \\
&= (z_2^{x^{-1}})^{y^{-1}} \bmod p = \\
&= ((z_1^y)^{x^{-1}})^{y^{-1}} \bmod p = \\
&= (((M^x)^y)^{x^{-1}})^{y^{-1}} \bmod p = \\
&= ((M^{xx^{-1}}) \bmod p)^{yy^{-1}}) \bmod p = \\
&= (M^{t(p-1)+1})^{s(p-1)+1} \bmod p = \\
&= (M^{t(p-1)} M)^{s(p-1)+1} \bmod p \overset{\text{МТФ}}{=} \\
&= M^{s(p-1)+1} \bmod p = M.
\end{aligned}
$$

Якщо M малого порядку, то задача дискретного логарифмування стає відносно простою, а отже можна знайти M, x або y. Щоб звести кількість елементів M : ord$M$ – мале, до неістотно малої, потріно, щоб в розкладі $p-1$ був великий, простий множник. В найкращому випадку p повинно бути сильнопростим ($p = 2p' + 1$, де p' – просте).

1.3 Схема шифрування Ель-Гамаля

Користувач А:

1. Вибирає велике, просте p, генератор $\alpha \in \mathbb{Z}_p^*$, секретний ключ k для розшифрування повідомлень відправлених до А.
2. $y = \alpha^k \bmod p$
3. Публікує відкритий ключ (p, α, y).

Зашифрування повідомлень $1 < M < p$ до А користувачем В:

1. Вибирає випадкове $x_M, 1 < x_M < p$.
2. Обчислює $C_1 = \alpha^{x_M} \bmod p, C_2 = y^{x_M} M \bmod p$
3. Формує ШТ (C_1, C_2) і відсилає до А.

Розшифрування повідомлення користувачем А:

$$(C_1^{-k} \bmod p) C_2 \bmod p = \alpha^{-x_M k} \alpha^{k x_M} M \bmod p = M$$

Стійкість схеми Ель-Гамаля базується на складності обчислення дискретного логарифму.

Зауваження. При шифруванні різних повідомлень M_i, M_j параметри x_{M_i}, x_{M_j} повинні бути різними.

Нехай криптоаналітик Е знайшов два ШТ (C_1, C_2) й (C_1', C_2') для шифрування яких використовувався один ключ x, що легко виявляється, так як $C_1 = C_1'$. Нехай також Е знає ВТ M. Тоді Е може знайти й M':

$$y \bmod p = {C_2}^{-1} \bmod p, M' = C_2' y^{-x} \bmod p = C_2'(C_2 M^{-1})^{-1} \bmod p = C_2' C_2^{-1} M \bmod p$$

Нехай криптоаналітик Е знайшов два ШТ (C_1, C_2) й (C_1', C_2') для шифрування яких використовувався один ключ x, що легко виявляється, так як $C_1 = C_1'$. Нехай також Е знає ВТ M. Тоді Е може знайти й M' :

$$y^x = C_2 M^{-1} \bmod p,$$

$$M' = C_2' y^{-x} \bmod p = C_2'(C_2 M^{-1})^{-1} \bmod p = C_2' C_2^{-1} M \bmod p$$

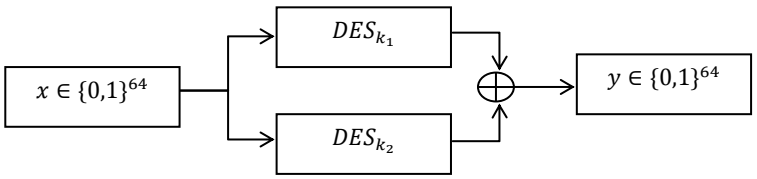
Атака імітації

Криптоаналітик Е будує криптоситсему шифрування Ель-Гамаля з відкритим ключем (p,α,y) і секретним ключом k. Публікує відкритий ключ (p,α,y), як відкритий ключ А. Тепер, всі повідомлення надіслані до А насправді прийдуть до Е. Тобто відкриті ключі ніяк не аутентифікуються.

1.4 Приклади односторонніх функцій в симетричній криптографії

1. Нехай $E_k(x)$ - алгоритм шифрування, стійкий до атаки на основі ВТ (тобто при відомій парі ВТ, ШТ неможливо знайти ключ). Функція $y = E_k(x_0) : \mathcal{K} \to \mathcal{Y}$, де $\mathcal{K}$ – множина ключів, $\mathcal{Y}$ – множина ШТ, x_0 – фіксоване. Функція E при цьому є обчислювально односторонньою.

2. Нехай задано блочний шифр (наприклад DES). Побудуємо функцію з фіксованими, відкритими ключами DES k_1 та k_2.



   Маємо односторонню функцію.

3. Розглянем функцію $y = f(k,M) = E_k(M) : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$. Якщо ШТ Y більше відстані єдиності, то по ШТ Y можно знайти ключ k і ВТ M. Але в загальному випадку, це можно зробити лише повним перебором, а отже маємо односторонню функцію з секретом.

1.5 Важкооборотні функції із секретом. Важкооборотна функція RSA

(Автор: Олександр Богуцький. Трохи редагувалось.)
(Версія від 19 січня 2017 р.)

Визначення 1.3. Важкооборотною функцією із секретом (лазівкою) називається відображення :

$$y = f_{\mathrm{k}}(x) : X \to Y,$$

що залежить від параметру k, таке, що:

1) $\forall k, x$ існує поліноміальний (ефективний) алгоритм обчислення $y = f_{\mathrm{k}}(x)$. При цьому знати k необов'язково.

2) При невідомому k майже для будь-яких k, y не існує поліноміального алгоритму обчислення оберненої функції $f_{\mathrm{k}}^{-1}(y)$.

3) При відомому k існує поліноміальний алгоритм обчислення $f_{\mathrm{k}}^{-1}(y)\ \forall k, x$.

Під ефективністю алгоритму розуміємо те, що обчислення ведуться за долі секунди. У той час як обчислення неефективних алгоритмів займатимуть мільярди років, а то й більше.

Зауважимо, що функція $y = f_{\mathrm{k}}(x)$ не обов'язково повинна бути бієкцією.

Як і раніше, потужності X, Y дуже великі.

Визначення 1.4. Важкооборотною функцією із таємницею RSA називається функція

$$y = x^e \bmod n, \tag{1.1}$$

при чому $n = p \cdot q$, $q \neq p$ — великі прості числа, $(e, \varphi(n)) = 1$. Зазвичай $p = 2p\prime + 1$, $q = 2q\prime + 1$, де $p\prime$, $q\prime$ — прості.

Секрет функції RSA: $q,\ p$, а також $\varphi(n)$.

Розглянемо оцінки складності:

1. Для будь-якого x складність обчислення (1.1) : $L_1 \leq 2\log n$.

2. При невідомих $q,\ p,\ \varphi(n)$ складність обчислення оберненої функції (тобто добування дискретного кореня): $L_2 = O(\sqrt{n})$.

Доведено субекспоненційну оцінку складності обернення фунції

$$L_3 = exp\{(\mathrm{c}_0 + O(1))\ln^{1/2} n(\ln\ln n)^{1/2})\}.$$

Крім того, вважається, але строго не доведено, що можна досягти такої складності обчислення $L_4 = exp\{(\mathrm{c}_0 + O(1))\ln^{1/3} n(\ln\ln n)^{2/3})\}$.

3. При відомих p, q складність обернення $L_5 \leq 2\log n$.

Помітимо, що якщо $x \in X = \{0, 1, ..., n-1\}$, то функція RSA (1.1) — бієкція.

Тепер розглянемо, як будується криптосистема RSA з використанням функції (1.1).

1.5.1 Побудова криптосистеми шифрування та цифрового підпису RSA користувачем A

1. Вибір великих простих p, q, $p \neq q$.

2. Обчислення $n = p \cdot q,\ \varphi(n) = (p-1)(q-1)$.

3. Знаходження $e:\ 1 < e < \varphi(n)$, $(e,\varphi(n)) = 1$.

4. Обчислення d із рівняння $d \cdot e = 1 \bmod \varphi(n)$, тобто $d = e^{-1} \bmod \varphi(n)$.

5. Оголошення відкритого ключа (n,e) для зашифрування повідомлень для користувача A та для перевірки цифрового підпису A. При цьому d — це секретий ключ A для розшифрування шифр текстів, відправлених користувачеві, і для формування цифрового підпису A.

Зрозуміло, що $\varphi(n)$ – теж секрет A.

Якщо $\varphi(n)$ відомо, то криптоаналітик складає систему рівнянь з двома невідомими:

$$\begin{cases} n = p \cdot q, \\ (p-1)(q-1) = \varphi(n) \end{cases},$$

та розв'язує її.

Обернена до RSA функція обчислюється легко :

$$x = y^d \bmod n.$$

1.5.2 Зашифрування повідомлення користувачем B

Зашифрування повідомлення M $(1 < M < n)$ відбувається наступним чином. Користувач, використовуючи відкритий ключ користувача A, обчислює :

$$C = M^e \bmod n.$$

Отриманий шифр текст C відправляється користувачеві A.

1.5.3 Розшифрування шифртексту С користувачем А

Розшифрувати шифр текст може лише користувач A, який знає секретний ключ. Здійснюється це теж всього лиш в одну операцію :

$$M = C^d \bmod n. \tag{1.2}$$

Дане твердження вимагає доведення, оскільки (1.2) не є очевидним фактом.

Доведення. Для доведення (1.2) розглянема 3 випадки:

1) $(M,n) = 1$.

2) $(M,n) > 1,\ M \vdots p,\ M \not\vdots q$.

3) $(M,n) > 1,\ M \not\vdots p,\ M \vdots q$.

На практиці, можливий лише перший випадок, оскільки ймовірність того, що M та n не взаємопрості, порядку $1/2^{500}$ (для випадку коли n–1024 біти), що практично дорівнює нулеві. Звісно, якби така ймовірність була високою, то здійснювалась би атака на взлом. Вибираючи M, шукали б НСД, що не дорівнює 1. Тоді за алгоритмом Евкліда, знаходяться p та q.

Розглянемо перший випадок: $(M,n) = 1$.

Тоді $C^d \bmod n = M^{ed} \bmod n$.

Оскільки $ed = 1 \bmod \varphi(n)$, то існує таке ціле число t, що $ed = t \cdot \varphi(n) + 1$.

Тому $M^{ed} \bmod n = M^{t\varphi(n)+1} \bmod n = (M^{\varphi(n)})^t \cdot M \bmod n$.

Оскільки за теоремою Ейлера $M^{\varphi(n)} \bmod n = 1$, то $(M^{\varphi(n)})^t \cdot M \bmod n = M$.

Випадок 2.

Розглянемо $C^d \bmod q = M^{ed} \bmod q = M^{(q-1)(p-1)\cdot t+1} \bmod q = (M^{q-1})^{(p-1)t} M \bmod q$.

Оскільки $M \not\vdots q$, то $M^{q-1} \bmod q = 1$ за малою теоремою Ферма.

Отримуємо $(M^{q-1})^{(p-1)t} \cdot M \bmod q = M$. $C^d \bmod p = M^{ed} \bmod p$. Оскільки $M \vdots p$, то $M^{ed} \bmod p = M \bmod p$. У результаті отримуємо співвідношення:

$$\begin{cases} C^d \bmod p = M, \\ C^d \bmod q = M \end{cases}.$$

За китайською теоремою про лишки, отримуємо $C^d \bmod n = M$.

Міркування щодо випадку 3 аналогічні міркуванням випадку 2. □

1.5.4 Задачі цифрового підпису

Основними задачами цифрового підпису є:

1. Підтвердження справжності (цілісності) повідомлення.
2. Підтвердження справжності джерела (автора) повідомлення.
3. Забезпечення неможливості підробки цифрового підпису.
4. Забезпечення неможливості відмови від цифрового підпису.
5. Можливість багакратної перевірки цифрового підпису різними користувачами без зміни системи цифрового підпису.
6. Юридична значимість.

1.5.5 Цифровий підпис RSA

Цифровий підпис формується просто. Користувач використовує власний секретний ключ для підпису повідомлення:

$$S = M^d \bmod n.$$

Далі відбувається формування підписаного повідомлення: до M дописується цифровий підпис S. Отже, підписане повідомлення виглядає так — (M, S).

Перевірка цифрового підпису користувачем B здійснюється наступним чином: отримане підписане повідомлення (M, S) розбивається за формальними ознаками на M та S, і перевіряється рівність

$$S^e \bmod n = M. \tag{1.3}$$

Якщо (1.3) виконується, то цифровий підпис вірний, тобто повідомлення справжнє, відправити та сформувати його міг лише користувач A і ніхто інший. При наявності іншого результату, це свідчитиме про змінене повідомлення або підроблений цифровий підпис. Якщо нема спотворень, то (1.3) виконується, оскільки $S^e \bmod n = M^{ed} \bmod n = M$ (доведення аналогічне попередньому).

Стійкість системи RSA базується на складності задачі факторизації (розклад великих чисел на множники). Якщо модуль n розкладений на множники q та p, то можна обернути важкооборотну функцію та взломати RSA, підробити цифровий підпис.

Не доведено, що якщо за шифр текстом в RSA знайдено відкритий текст, то можна розкласти n на множники p та q.

1.6 Складність алгоритмів

(Автор: Антон Вихло. Не редагувалось.)
(Версія від 18 січня 2017 р.)

Алгоритм – загальна послідовна процедура, яка виконує покрокове рішення певної задачі.

Вхід алгоритму – деяка скінченна множина, представлена у визначеній системі кодування даних.

Розмір даних можна оцінювати як число символів вхідних даних у певній системі кодування.

Після зупинки алгоритму, результат роботи представляється набором символів в певній системі кодування.

Часова складність алгоритму визначається числом кроків до зупинки(або часу, наприклад, роботи ЕОМ).

Алгоритм – дискретна процедура, в якій проміжні та кінцеві результати змінюються покроково.

Ємнісна складність алгоритму оцінюється максимальною кількістю символів, що оброблюються на кожному кроці. При реалізаціїї алгоритмів, наприклад, на ЕОМ, максимальний об'єм необхідної пам'яті.

Формальні теорії алгоритмів

1. Детерміновані машини Т'юрінга(ДМТ), недетерміновані машини Т'юрінга(НДМТ).
2. Рекурсивні функції.
3. Нормальні алгоритми Маркова.
4. Теория Поста.

Узагальнений тезис Черча

Будь-який интуітивно зрозумілий алгоритм може бути сформульований в будь-якій із теорій 1-4.

Позначення:

– A_n – алгоритм.

– X – область визначення алгоритму(вхідні дані, що належать X).

– $x \in X, x$ – індивідуальний вхід(дані).

– $||x||$ – розмір входу.

– $T(n,x), x \in X, n = ||x||$ – часова складність алгоритму A_n .

– $S(n,x), x \in X, n = ||x||$ – ємнісна складність алгоритму A_n .

Середньою складністю називається

$T_{\text{сер.}}(n) = \sum\limits_{x \in X, ||x||=n} p(x) \cdot T(n,x), x \in X, n = ||x||, p(x)$ - деякий розподіл імовірностей на множині X.

Складністю в найгіршому випадку називається

$T_{max}(n) = \max\limits_{x \in X, ||x||=n} T(n,x)$.

Асимптотичні оцінки складності

Складність для майже всіх входів називається така оцінка $T_{\text{м.в.}}(n)$, що $\forall \epsilon \geq 0\ \exists n_0$ така, що відносна частина тих входів $x \in X, n = ||x||$, для яких
$T(n) \geq T_{\text{м.в.}}(n) \to 0$, при $n \to \infty$.

Аналогічно визначається $S_{\text{сер.}}(n), S_{max}(n), S_{\text{м.в.}}(n)$.

Визначення 1.5. Поліноміальною часовою складністю називається така $T(n)$, що
$\exists p(x)$ – поліном, і $T(n) = \mathcal{O}(p(n)), n \to \infty$, де під $T(n)$ розуміється будь-яка складність $T_{\text{сер.}}(n), T_{max}(n), T_{\text{м.в.}}(n)$.

Визначення 1.6. Експоненційною часовою складністю називається $T(n) \asymp a^{cn}$,
де $a = const \geq 1$, c = const >0,$n \to \infty$.

Інше означення $T(n) = \theta(a^{cn})$.
$T(n) \asymp a^{cn} \iff T(n) = \mathcal{O}(a^{cn})$ і $a^{cn} = \mathcal{O}(T(n))$
Інакше $\exists 0 \leq c_1 \leq c_2 \leq \infty$ такі, що $c_1 a^{cn} \leq T(n) \leq c_2 a^{cn}$
Зазвичай 0 <c <1, $a = 2; e; 10$.

Визначення 1.7. Субекспоненційною оцінкою складності (часової) називається така оцінка $T(n) \asymp \exp(n^{\gamma}(\ln n)^{1-\gamma})$, $0 \leq \gamma \leq 1, n \to \infty$.

При $\gamma \to 0$ субекспоненційна оцінка наближається до поліноміальної.
При $\gamma \to 1$ субекспоненційна оцінка наближається до експоненціальної.
Аналогічно визначаються поліноміальні, експоненційна та субекспоненційні оцінки $S(n)$.

Визначення 1.8. Класом P називається клас задач для яких $\exists$ поліноміальний часовий алгоритм (в найгіршому випадку) на ДМТ рішення задачі.

Визначення 1.9. Класом NP називається клас задач для яких $\exists$ поліноміальний часовий алгоритм (в найгіршому випадку) на НДМТ рішення задачі.

Відомо: $P \subseteq NP$.
Не доведено: $P = NP$ або $NP \backslash P \neq \varnothing$.

Розділ 2

Криптографічні геш-функції та схеми цифрового підпису

2.1 Геш-функції

(Автор: Євген Грубіян. Не редагувалось.)
(Версія від 18 січня 2017 р.)

Геш-функція - одностороння функція, яка перетворює блок даних довільного розміру в блок даних фіксованого розміру. При цьому прообразів для одного значення геш-функції може бути дуже багато, а знаходження бодай одного із них є дуже трудомістким. Однією із причин розробки геш-функцій стала необхідність підписувати повідомлення довільного розміру та перевіряти цілісність даних.

Геш-функції в криптографії використовується:

1. Для створення геш-образів для повідомлень цифрового підпису.
2. Для безпечного зберігання паролів.
3. Для автентифікації.
4. В криптопротоколах.
5. Для генерації випадкових послідовностей, функцій, ключів.
6. Для перевірки правильності обчислювальних операцій.

Розглянем варіант цифрового підпису *RSA* без геш-функцій:
Нехай абонент **A** має криптосистему *RSA* з відкритим ключем (n, e) і таємним ключем d.

Алгоритм 2.1 (Цифровий підпис *RSA* без геш-функцііїї)**.**

- **A** розбиває M на блоки: $M = M_1 M_2 \dots M_t$, де $|M_i| < n$, $i = \overline{1, t}$.
- **A** обчислює цифровий підпис для кожного із блоків окремо $S_i = M_i^d \ mod \ n$
- **A** формує підписане повідомлення $(M, S_1 S_2 \dots S_t)$

Недоліки цього способу:

1. Трудоємність.
   Очевидно, що чим більше блоків необхідно підписати тим більше часу потрібно витратити, тому даний варіант є не практичним з обчислювальної точки зору.

2. Атаки перестановкою і видаленням блоків.
   Оскільки блоки підписуються незалежно криптоаналітик **E** може видаляти та переставляти блоки як йому завгодно без порушення правильності підпису, продукуючи повідомлення з вірним цифровим підписом, які **A** ніколи не підписував і не підписав би.

3. Комутативні атаки.
   Нехай A підписав два повідомлення $(M_1,\, S_1)\,,\ (M_2,\, S_2)$. Криптоаналітик **E** формує неправдиве повідомлення з вірним цифровим підписом $(M_3,\, S_3)$, де $M_3 = M_1M_2 \ mod\ n$, $S_3 = S_1S_2 \ mod\ n$. Перевірка цифрового підпису цього повідомлення дає

$$S_3^e \ mod\ n = (S_1S_2)^e \ mod\ n = (M_1^dM_2^d)^e \ mod\ n = M_1^{de}M_2^{de} \ mod\ n = M_1M_2 \ mod\ n = M_3$$

   Тобто підпис правильний. І взагалі кажучи $\forall(M_i,\, S_i)$, $i = \overline{1,\, t}$ повідомлень, які підписав **A** можна скласти неправдиве повідомлення з вірним цифровим підписом: $(\widetilde{M},\, \widetilde{S})$, де

$$\widetilde{M} = \left(\prod_{i=1}^{t} M_i^{\alpha_i}\right) mod\ n,\ \widetilde{S} = \left(\prod_{i=1}^{t} S_i^{\alpha_i}\right) mod\ n,\ \ \alpha_i \in \mathbb{Z}_+$$

4. Атака з неявним використанням таємного ключа d
   Нехай криптоаналітик **E** хоче щоб **A** підписав повідомлення M, яке вигідне криптоаналітику.

   Алгоритм 2.2 (Атака з неявним використанням таємного ключа)**.**

   – **E** вибирає деяке число r: $gcd(r,n) = 1$.
   – **E** обчислює $\widetilde{M} = r^eM \ mod\ n$.
   – **E** відправляє повідомлення M для цифрового підпису **A**.
   – Якщо **A** підписав запропоноване повідомлення $\widetilde{M}$, **E** отримує підписане повідомлення $(\widetilde{M},\, \widetilde{S})$ та може обчислити підпис початкового вигідного для **E** повідомлення M:
$$\widetilde{S} = \widetilde{M}^d \ mod\ n = r^{ed}M^d \ mod\ n = rM^d \ mod\ n$$
$$S = M^d = \widetilde{S}r^{-1} \ mod\ n$$
   – **E** формує вигідне для нього підписане повідомлення $(M,\, S)$.

Введемо позначення:
$\mathbb{Z}_2 = 0, 1$ - алфавіт із двох символів.
$\mathbb{Z}_2^*$ - замикання Кліні алфавіту $\mathbb{Z}_2$ (Множина послідовностей всіх довжин із алфавіту). $\mathbb{Z}_2^m$ - множина послідовностей довжини m.

Визначення 2.1. *Геш-функцією* в криптографії називається відображення:

$$h\colon \mathbb{Z}_2^* \to \mathbb{Z}_2^m,\ m > 1$$

яке має властивості:

1. $\forall M \in \mathbb{Z}_2^*$ обчислення геш-образу $H = h(M)$ швидке.

2. $h(x)$ - обчислювально одностороння функція, тобто $\forall H \in \mathbb{Z}_2^m$ неможливо знайти хоча б одне M, таке що $H = h(M)$.
   Приклад

$m = 256$, $||M|| = 500$. Тоді в середньому $\forall H \in \mathbb{Z}_2^m \ \exists \frac{2^{500}}{2^{256}} = 2^{244}$ можливих M. Тобто знайти таке повідомлення-прообраз випадково неможливо.

3. Геш-функція чутлива до змін входу, тобто має лавинні ефекти.

4. Слабка стійкість до колізій, тобто для кожного фіксованого повідомлення M неможливо знайти таке $M' \neq M$ таке, що $h(M) = h(M')$.

5. Сильна стійкість до колізій, тобто неможливо знайти хоча б одну пару повідомлень (M, M'), таку що $h(M) = h(M')$

Визначення 2.2. Геш-функція, яка має властивості 1-4 називається *слабкою односторонньою геш-функцією.*

Визначення 2.3. Геш-функція, яка має властивості 1-5 називається *сильною односторонньою геш-функцією.*

Розглянемо цифровий підпис *RSA* до повідомлення M будь-якої довжини з геш-функцією $h(x)$. Нехай абонент **A** має криптосистему *RSA* з відкритим ключем (n, e), таємним ключем d і відкриту геш-функцією $h(x)$: $||h|| = m$, $2^m < n$

Зауваження. Зазвичай довжина виходу геш-функції m складає 128, 160, 192, 256 або 512 біт. Найпоширенішими є геш-функції з m 128, 160, 256. На цих довжинах досягається компроміс між стійкістю до колізій та швидкістю обчислення геш-функції.

Алгоритм 2.3 (Цифровий підпис *RSA* з геш-функцією)**.**

- **A** обчислює значення геш-функції повідомлення M: $H = h(M)$
- **A** обчислює цифровий підпис: $S = H^d \ mod \ n$
- **A** формує підписане повідомлення (M, S).

Нехай підписане повідомлення (M, S) перевіряє абонент **B**.

Алгоритм 2.4 (Перевірка цифрового підпису *RSA* з геш-функцією)**.**

- **B** розбиває (M, S) на M та S по формальним ознакам
- **B** обчислює $H = h(M)$
- **B** перевіряє рівність $S^e \ mod \, n = H$. Якщо вона виконується, то цифровий підпис *вірний*, якщо не виконується, то підпис *не вірний.*

Розглянемо можливості атак на цифровий підпис з геш-функцією. Основні атаки базуються на пошуці колізій в використаній геш-функції, тобто фактів однакових геш-значень при різних аругументах геш-функції.
Нехай криптоаналітик **E** намагається отримати вигідне для нього повідомлення з вірним цифровим підписом абонента **A**. Розглянемо наступні два варіанта атаки:

1. Якщо **E** має підписане **A** повідомлення (M, S)
   E будує ряд повідомлень $M_1, M_2, \ldots, M_t$: $M_i \neq M$ та обчислює їхні геш-значення $H_1, H_2, \ldots, H_t$: $H_i = h(M_i)$.
   Пободуємо ймовірнісну модель атаки, в припущенні, що всі геш-значення рівноймовірні на множині $\mathbb{Z}_2^m$. Позначимо P_1, Q_1 - ймовірності слабкої колізії і відсутності слабкої колізії відповідно. Тоді:

$$Q_1 = \left(1 - \frac{1}{2^m}\right)^t, \ P_1 = 1 - \left(1 - \frac{1}{2^m}\right)^t$$

Таким чином асимптотично можна записати:

$$Q_1 = e^{-t/2^m},\ P_1 = 1 - e^{-t/2^m}\ m,t \to \infty$$

Середня кількість необхідних повідомлень для здійснення атаки знаходженням колізії:

$$t = 2^m ln\frac{1}{1-P_1}$$

2. Якщо криптоаналітик **E** не чекає підписаного **A** повідомлення(*атака Ювала*).
E будує ряди повідомлень прийнятних $M_1, M_2, \ldots, M_t$ та неприйнятних $\widetilde{M_1}, \widetilde{M_2}, \ldots, \widetilde{M_t}$ для **A** повідомлень, тобто таких, які він би підписав і не підписав ніколи відповідно та обчислює їхні геш-значення $H_1, H_2, \ldots, H_t$ та $\widetilde{H_1}, \widetilde{H_2}, \ldots, \widetilde{H_t}$ відповідно. Якщо знайдеться така пара індексів $(i,\ j)$, що $H_i = \widetilde{H_j}$, то криптоаналітик **E** посилає повідомлення $(M_i,\ H_i)$ для цифрового підпису **A** та отримавши підписане повідомлення $(M_i,\ S_i)$ формує неправдиве повідомлення з вірним цифровим підписом $\left(\widetilde{M_j},\ S_i\right)$.
Будуючи ймовірнісну модель по аналогії з попереднім випадком варто зазначити, що в якості ймовірностей P_2, Q_2 варто брати ймовірності сильної колізії та її відсутності:

$$Q_2 = \left(1 - \frac{1}{2^m}\right)^{t^2},\ P_2 = 1 - \left(1 - \frac{1}{2^m}\right)^{t^2}$$

Варто зазначити що при однаковому t ймовірність сильної колізії більша за ймовірність слабкої.
Середнє число повідомлень для успішної атаки складатиме:

$$t = 2^{m/2}\sqrt{ln\frac{1}{1-P_2}}$$

2.2 Загальні схеми побудови геш-функцій. Характеристики відомих геш-функцій. Цифровий підпис Ель-Гамаля.

(Автор: Яна Євсюкова. Не редагувалось.)
(Версія від 18 січня 2017 р.)

2.2.1 Загальні схеми побудови геш-функцій.

Геш-функція необхідна для того, щоб перетворювати послідовність будь-якої довжини в зазначену. Існує декілька схем побудови геш-функції:

Схема l. Побудова геш-функції

Нехай M-відкритий текст. Відкритий текст розбивається на блоки $M = M_1 M_2 \ldots M_t$. Довжина M_i-ого блоку дорівнює m біт, де $i = \overline{1,t}$.
Будується функція стиснення. Зазвичай дана функція є односторонньою, що володіє лавинним ефектом та іншими характеристиками. Причому можна брати вже відомі нам односторонні функції. Наприклад, функція Діффі-Гелмана, RSA, тобто ті функції, що використовуються в асиметричній криптографії з відкритими ключами. Однак, в стандартних геш-функціях, вони зазвичай не використовуються, оскільки працюють досить повільно.

Функції стиснення будуються завдяки деяким стандартним операціям: *XOR, додавання за модулем, циклічний зсув, логічні операції* та інші.
Наприклад,

$$(n,m) - f$$

функція, на вході якої n біт, а на виході m біт.
Далі, по рекурентній процедурі обчислюється

$$H_i = F(M_i \oplus H_{i-1}), i = \overline{1,t}$$

де H_0 початковий вектор (зазвичай, відкритий).
Останній блок H_t - це вихідне значення геш-функції $h(x)$.

$$H_t = h(M)$$

Тобто дана процедура є покрокова. На кожному кроці оброблюється один блок відкритого тексту M та весь час замішуються попередні результати з наступними блоками. І в останньому значенні H_t будуть замішані всі біти повідомлення.
Існує й інша процедура гешування. Наприклад,

$$(2m,m) - f$$

функція, на вході якої $2m$ біт, а на виході m біт.
Тоді функція H_i будується наступним чином:

$$H_i = F(M_i, H_{i-1}), i = \overline{1,t}$$

Схема ll. Побудова геш-функції

Нехай M-відкритий текст. Відкритий текст розбивається на блоки $M = M_1 M_2 \dots M_t$, $\|M_i\| = m$ біт, $i = \overline{1,t}$.
Нехай, відповідно $E_k(x)$-алгоритм блочного шифрування. При цьому розмір входу, виходу та ключа однакові (m біт).

$$\|E_k\| = \|k\| = \|x\| = m$$

Тоді, відповідно, рекурентно обчислюється:

$$H_i = E_A(B) \oplus C, i = \overline{1,t}$$

де A,B,C-m-вимірні вектора.
A,B,C можуть приймати одне із значень:

1. значення блоку M_i
2. значення попереднього гешу H_{i-1}
3. $M_i \oplus H_{i-1}$
4. $D = const$ - m-вимірний вектор

Усього отримаємо 64 можливих варіанти (на кожен із трьох векторів маємо по чотири можливих значення, тобто $4^3 = 64$). 52 з них признані слабкими, тому використовують лише 12 останніх варіантів.
Наприклад,

1. $H_i = E_{M_i}(M_i) \oplus H_{i-1}$
2. $H_i = E_{H_{i-1}}(M_i) \oplus M_i$
3. $H_i = E_{M_i}(M_i \oplus H_{i-1}) \oplus H_{i-1}$
4. $H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i$

Вихідним значенням геш-функції $h(M)$ є останній блок.

$$h(M) = H_t$$

Характеристики відомих геш-функцій.

Розглянемо випадок, коли геш-функції побудовані по **Схемі I**, тобто, коли використовується не шифратор, а лише бінарні операції.

1. Автором перших геш-функцій,що використовувались, став Рівест (один з авторів RSA). Ним були створені такі геш функції:

   MD2 (1989р.)
   MD4 (1990р.)
   MD5 (1991р.)

   Вихідна строка даних геш-функцій складає 128 біт, що зараз є недодтатнім для використання.
   Геш-функція *MD2* майже не використовувалась. Вона створена для 8-розрядних процесорів, а *MD4* та *MD4* для 32-розрядних процесорів.
   При цьому, коли запропонували геш-функцію *MD4*, майже одразу було знайдено вразливості з точки зору побудови колізій. Тому Рівест трохи ускладнив функцію та отримав *MD5*. Вона працює повільніше, ніж *MD4*, однак знайдена колізія була усунена. Після чого *MD5* років 20 була найпопулярнішою геш-функцією, але зараз вона знімається з використання.

2. *SHA-1* - стандарт США (1995р).
   SHA-1 використовувалсь у стандарті цифрового підпису *SHS*.
   Вихідна строка даної геш-функцій складає 160 біт.
   Функції стиснення в пункті 1 та пункті 2 будувалися чергуванням операцій XOR, циклічного сзуву, додавання за модулем 2^{32}, логічних операцій AND та OR. Чергуя дані операції, будувалась функція, яка обчислювально була одностороння та володіла гарними властивостями перемішування.

3. Переможець конкурсу *SHA-3* (2007-2012рр.).
   В 2010 році на конкурсі було відібрано 14 кандитатів на перемогу. А в 2012 році проголосили переможця. Ним став *Keccak* з вихідною строкою у 256 біт.

Розглянемо випадок, коли геш-функції побудовані по **Схемі II**, тобто, коли використовується шифратор.

1. *ГОСТ Р 34-11* (1994р.).
   В Україні є стандартом з 1995 року. Також є стандартом Россії та деяких держав СНД.
   Вихідна строка даної геш-функцій складає 256 біт.
   Для побудови функції стиснення використовується шифр *ГОСТ Р 28147-89*.
   Довгочасні ключі фіксуються(декілька варіантів на вибір).

Таким чином, усі зазначені геш-функції є функціями без секрету, тобто кожен може використовувати їх.

Цифровий підпис Ель-Гамаля.

Більшість європейских держав будують сучасний цифровий підпис за схемою Ель-Гамаля. Розглядатиметься базовий варіант підпису Ель-Гамаля, в той час як існують сотні варіантів даного цифрового підпису. Вони мають однакову односторонню функцію що визначає стійкість, а саме функцію Діффі-Геллмана. Стійкість цифрового підпису Ель-Гамаля побудована на складності залачі дискретного логарифмування. Побудова підпису подібна до схеми шифрування Ель-Гамаля.

Побудова схеми цифрового підпису Ель-Гамаля користувачем А:

1. **А** обирає велике просте число p та примітивний елемент α поля F_p, де p та α не є секретними.

2. **А** обирає секретний ключ k та обчислює

$$y = \alpha^k mod p$$

   Також обирається геш-функція $h(x)$.

3. **А** оголошує відкритий ключ (p,α,y) для перевірки цифрового підпису. Секретний ключ k створений для формування цифрового підпису.

Формування цифрового підпису до повідомлення M довільної довжини з геш-функцією користувачем А

1. **А** обирає випадковий параметр цифрового підпису x_M (для кожного попідомлення M параметр обирається знову), $1 < x_M < p-1$, $\text{gsd}(x_M, p-1) = 1$.
   Обчислюється

$$r = \alpha^{x_M} mod p$$

2. Знаходимо розв'язок рівняння відносно S:

$$H = kr + x_M S mod(p-1)$$

   де $H = h(M)$, тобто

$$S = (H - kr)x_M^{-1} mod(p-1)$$

3. **А** формує підписане повідомлення (M,r,S) де (r,S) є цифровим підписом.

Перевірка цифрового підпису користувачем В

1. Користувач **В** ділить повідомлення на три частини:

$$(M,r,S) \to M,r,S$$

2. Обчислює $h(M) = H$

3. Перевіряє рівність:

$$?y^r r^S mod p = \alpha^H ?$$

   Якщо рівність виконується, то підпис вірний, і навпаки, якщо рівність не виконується, то підпис хибний.

Дійсно, якщо спотворення відсутні, то виконується:

$$y^r r^S mod p = \alpha^{kr+x_M S} mod p = \alpha^{t(p-1)+H} mod p = (\alpha^{p-1})^t \alpha^H mod p = \alpha^H$$

де $(\alpha^{p-1}) mod p \equiv 1$ за малою теоремою Ферма.

Зауваження! Під час формування цифрових підписів різних повідомлень M_1 та M_2, параметри цифрового підпису x_{M_1} та x_{M_2} не мають співпадати.
Нехай $x_{M_1} = x_{M_2}$. Криптоаналітик може виконати успішну атаку, оскільки він має:

$$(M_1,r_1,S_1) = (M_2,r_2,S_2)$$

Криптоаналітик помічає, що $r = r_1 = r_2 = \alpha^x mod p$, тобто $x_{M_1} = x_{M_2}$ та складає систему рівнянь:

$$h(M_1) = H_1 = kr + xS_1 mod(p-1)$$

$$h(M_2) = H_2 = kr + xS_2 mod(p-1)$$

Відповідно, криптоаналітик знаходить секретний ключ k та параметр x. Таким чином, система повністю зламана.

2.3 Одностороння функція Рабіна з секретом

(Автор: Антон Карпець. Трохи редагувалось.)
(Версія від 19 січня 2017 р.)

2.3.1 Визначення

Визначення 2.4. Односторонньою функцією Рабіна називається функція:

$$y = x^2 \bmod n, \tag{2.1}$$

де $n = p \cdot q$, $p \neq q$ — великі прості числа. Секретом функції Рабіна є числа p та q.

Зауваження. Одностороння функція Рабіна не є частковим випадком односторонньої функції RSA ($y = x^e \bmod n$, $n = p \cdot q$, $p \neq q$ — великі прості числа), оскільки в схемі RSA $(e, \varphi(n)) = 1$, тобто, $(e, (p-1)(q-1)) = 1$. Це означає, що e не ділиться націло на 2.

Зауваження. Функція Рабіна не є бієкцією на множині $X = \{0, 1, ..., n-1\}$ на відміну від RSA, яка є бієктивним відображенням на множині X. Окрім того, значення даної функції є квадратичними лишками, якщо $(x, n) = 1$.

Зауваження. Функція, обернена до функції (2.1) - $y^{\frac{1}{2}} \bmod n$, яка називається дискретним добуванням квадратного кореня, обчислюється лише для тих y, які є квадратичними лишками і не є однозначною.

2.3.2 Складність обчислень

1. Для $\forall x \in X = \{0, 1, ..., n-1\}$ обчислення $y = x^2 \bmod n$ потребує однієї операції піднесення до квадрату за модулем n.

2. Секретом функції Рабіна є числа p та q. Тому, коли вони відомі, то обернення функції (2.1) (отримання значення оберненої до y функції $y^{\frac{1}{2}} \bmod n$) обчислюється за

$$L_1 = O(\log n)$$

операцій піднесення до квадрату та множення за модулем n.

3. У випадку, коли розклад $n = p \cdot q$ невідомий, то складність обернення становить

$$L_2 = O(\exp(\ln \frac{1}{2} \cdot n \cdot (\ln \ln n)^{\frac{1}{2}})),$$

що є доведено, або

$$L_3 = O(\exp(\ln \frac{1}{3} \cdot n \cdot (\ln \ln n)^{\frac{2}{3}})),$$

що припускається.

Алгоритм 2.5. Обернення функції Рабіна при відомому розкладі $n = pq$

1. Обчислення $y^{\frac{1}{2}} \bmod p$, де p - просте число. Існує три можливі випадки:

- $p = 4m + 3$,
- $p = 8m + 5$,
- $p = 8m + 1$, де $m \in \mathbb{Z}$.

Нехай $p = 4m + 3$, y - квадратичний лишок. Тоді справедлива наступна рівність: $y^{\frac{p-1}{2}} = 1 \bmod p$, звідки маємо, зважаючи на вигляд числа p, що $y^{2m+1} = 1 \bmod p$. Домноживши на y та взявши квадратний корінь з обидвох сторін, маємо: $y^{\frac{1}{2}} = \pm y^{m+1} \bmod p$.

Нехай $p = 8m + 5$, y - квадратичний лишок. Тоді справедлива наступна рівність: $y^{\frac{p-1}{2}} = 1 \bmod p$, звідки маємо, зважаючи на вигляд числа p, що $y^{4m+2} = 1 \bmod p$. Добувши квадратний корінь, отримуємо: $y^{2m+1} = \pm 1 \bmod p$. У випадку, коли $y^{2m+1} = 1 \bmod p$, маємо: $y^{\frac{1}{2}} = \pm y^{m+1} \bmod p$.

Зауваження. 2 - квадратичний нелишок за модулем $p = 8m+5$, оскільки $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} = -1$. Тоді $2^{\frac{p-1}{2}} = 2^{4m+2} = -1 \bmod p$

У випадку, коли $y^{2m+1} = -1 \bmod p$, домноживши дане рівняння на $2^{4m+2} = -1 \bmod p$, отримуємо: $y^{2m+1} \cdot 2^{4m+2} = 1 \bmod p$. Таким чином, $y^{\frac{1}{2}} = \pm y^{m+1} \cdot 2^{2m+1} \bmod p$.

Нехай $p = 8m + 1$, y - квадратичний лишок. Тоді справедлива наступна рівність: $y^{\frac{p-1}{2}} = 1 \bmod p$, звідки маємо, зважаючи на вигляд числа p, що $y^{4m} = 1 \bmod p$, що рівносильно: $y^{2m} = \pm 1 \bmod p$. Далі шукається квадратичний нелишок a, який використовується декілька разів в рівняннях для отримання $(+1)$ в правій частині (аналогічно попередньому випадку).

Зауваження. Якщо $p = 4m+3$, то p називається простим числом Блюма. Якщо $n = p \cdot q, p = 4m + 3, q = 4k + 3$, то число n називається числом Блюма.

2. Обчислення $y^{\frac{1}{2}} \bmod q$, де q - просте число - аналогічно першому пункту.

3. Обчислення $y^{\frac{1}{2}} \bmod n$, де $n = p \cdot q, p \neq q$ (власне, обернення функції Рабіна)

Оскільки $p \neq q$, то $(p, q) = 1$. Тоді маємо систему:

$$\begin{cases} y^{\frac{1}{2}} = \beta_p \bmod p, \\ y^{\frac{1}{2}} = \beta_q \bmod q. \end{cases}$$

За китайською теоремою про лишки: $y^{\frac{1}{2}} \bmod n = [\pm\beta_p \cdot q \cdot (q^{-1} \bmod p) \pm \beta_q \cdot p \cdot (p^{-1} \bmod q)] \bmod n$.

Алгоритм 2.6. Факторизація n = pq при відомих чотирьох коренях функції Рабіна Маємо функцію Рабіна $y = x^2 \bmod n$, де $n = p \cdot q$, $p \neq q$. Нехай відомі її корені x_1, x_2, x_3, x_4. Знайдемо серед них такі два корені x_i та x_j, що $x_i \neq \pm x_j$. Маємо систему:

$$\begin{cases} x_i^2 = y \bmod n, \\ x_j^2 = y \bmod n \end{cases}.$$

Віднявши від верхньої частини нижню, маємо: $(x_i - x_j)(x_i + x_j) = 0 \bmod n$. Але $(x_i - x_j) \not\vdots n$ та $(x_i + x_j) \not\vdots n$. Отже, $(x_i + x_j, n) = p$ або $(x_i + x_j, n) = q$. За алгоритмом Евкліда знаходимо p або q та, відповідно, $q = \frac{n}{p}$ або $p = \frac{n}{q}$.

Твердження 2.1. *Задача добування квадратного кореня в функції Рабіна $y = x^2 \bmod n$, де $n = p \cdot q$, $p \neq q$, поліноміально еквівалентна задачі факторизації $n = pq$.*

2.4 Системи шифрування та цифровий підпис Рабіна

(Автор: Маша Коляда. Редагувалось(Свічкарьов))
(Версія від 19 січня 2017 р.)

Дана система шифрування та цифрового підпису використовує односторонню функцію Рабіна : $y = x^2 \bmod n$, де $n = p \cdot q$, $p \neq q$ – великі прості числа.

Побудова системи шифрування Рабіна користувачем A:

2.4.1 Схема шифрування Рабіна 1

Зашифрування відкритого тексту користувачем B

1. Користувач A обирає прості числа p,$q : p \neq q$. Зазвичай $p = 2 \cdot p' + 1$, $q = 2 \cdot q' + 1$, де p', q' – великі прості числа.
2. Користувач A обчислює $n = p \cdot q$, де p, q – його секретний ключ.
3. Користувач A *«оголошує»* відкритий ключ n для зашифрування повідомлень до нього.
4. Зашифрування повідомлення M користувачем B: $C = M^2 \bmod n$ – шифртекст, де $\sqrt{n} < M < n$.
5. Користувач B відправляє повідомлення користувачу A.

Зауваження: При $M^2 < n$, розрахунок квадратного кореня аналогічний до розрахунку квадратного кореня на множині дійсних чисел, що легко виконується.

Розшифрування шифртексту користувачем A:

1. Використовуючи p, обчислює: $\pm\beta_p = C^{\frac{1}{2}} \bmod p$.
2. Використовуючи q, обчисює: $\pm\beta_q = C^{\frac{1}{2}} \bmod q$.
3. За китайською теоремою про лишки знаходить 4 корені:

$$C^{\frac{1}{2}} \bmod n = (\pm\beta_p) \cdot q \cdot q^{-1} \bmod p + \beta_q \cdot p \cdot p^{-1} \bmod q.$$

Отримує M_1,M_2,M_3,M_4, після чого обирає один з цих коренів за змістом.

2.4.2 Схема шифрування Рабіна 2(коли n – число Блюма)

Зашифрування відкритого тексту M : $\sqrt{n} < M < n$ користувачем B

1. Обчислює $M^2 \bmod n = C$.
2. Обчислює 2 біти:

$$b_1 = \begin{cases} 0, & \text{якщо } M \text{ парне} \\ 1, & \text{інакше} \end{cases}$$

$$b_2 = \begin{cases} 0, & \text{якщо } \left(\frac{M}{n}\right) = -1 \\ 1, & \text{якщо } \left(\frac{M}{n}\right) = 1 \end{cases}$$

3. Формує зашифроване повідомлення (C,b_1,b_2) та відправляє його користувачу A.

Розшифрування (C,b_1,b_2) користувачем A

1. Аналогічно схемі шифрування Рабіна 1 знаходить M_1,M_2,M_3,M_4.
2. Використовуючи (b_1,b_2), обчислює $(b_1^i, b_2^i), i = \overline{1,4}$ та обирає M_i у якого $(b_1^i, b_2^i) = (b_1,b_2)$.

Стійкість до атаки на основі шифртексту повністю заснована на(еквівалентна) складності задачі факторизації. Схеми шифрування Рабіна 1 і 2 не стійкі до атаки на основі вибраного шифртексту.

2.4.3 Атаки на схеми шифрування

Атака на схему шифрування Рабіна 1

1. Криптоаналітик E підбирає послідовність відкритого тексту $M_1 \ldots M_k$, обчислює відповідні їм $C_1 \ldots C_k$.
2. Відправляє отриманий шифртекст A.
3. $A : M_1' \ldots M_k'$. Перевіряє $M_i = \pm M_i'$.
4. Якщо така пара знайшлась, то обчислює $(M_i \pm M_i'$,$n) = p$ або q.

Атака на схему шифрування Рабіна 2

1. Криптоаналітик E обирає відкритий текст M. Обчислює (b_1,b_2).
2. Обчислює $M^2 \bmod n = C$.
3. Відправляє шифртекст (C, b_1, b_2'), де $b_2' = b_2 \oplus 1$
4. Отримавши відкритий текст M', знає, що $M \neq M'$, обчислює $\left(M_i \pm M_i'$,$n\right) = p$ або q.

2.4.4 Схема цифрового підпису Рабіна

Побудова схеми цифрового підпису Рабіна користувачем А

1. Обирає великі прості числа p,q : $p \neq q$.
2. Обчислює геш-функцію $h(x)$.
3. Оголошує $n = p \cdot q$.
4. Оголошує відкритий ключ $(n, h(x))$ для перевірки цифрового підпису A.

При цьому p,q – секретний ключ користувача A для формування цифрового підпису.

Формування цифрового підпису А

1. $M = m_1 m_2 \dots m_k$. Генерує випадкову послідовність $R = r_1 \dots r_s$.
2. Обчислює $h(M||R) = h(m_1 \dots m_k + r_1 \dots r_s) = H$
3. Перевіряє, чи є H квадратичним лишком за модулем n.
   (a) Якщо виконується $H^{\frac{p-1}{2}} \bmod p = 1$ і $H^{\frac{q-1}{2}} \bmod q = 1$, переходимо до кроку 4.
   (b) інакше переходимо до кроку 1.

   Якщо n–число Блюма, то в средньому, цей пункт виконується на 4-ій спробі.
4. Обчислює $H^{\frac{1}{2}} \bmod n$:
   (a) Використовуючи p обчислює: $\pm\beta_p = C^{\frac{1}{2}} \bmod p$
   (b) Використовуючи q обчислює: $\pm\beta_q = C^{\frac{1}{2}} \bmod q$
   (c) За китайською теоремою про лишки знаходить 4 корені:

$$C^{\frac{1}{2}} \bmod n = \pm\beta_p \cdot q \cdot q^{-1} \bmod p + (\beta_q) \cdot p \cdot p^{-1} \bmod q \rightarrow M_1,M_2,M_3,M_4$$

5. Обирає один з коренів β, та формує підпис (M,R,β)

Перевіка цифрового підпису користувачем В

1. $(M,R,\beta) \rightarrow M,R,\beta$
2. Обчислює $h(M||R) = h(m_1 \dots m_k + r_1 \dots r_s) = H$
3. Перевіряє рівність : $\beta^2 = H \bmod n$.
   (a) Якщо виконується, то підпис правильний
   (b) Інакше, не правильний.

Складність взлому цифрового підпису Рабіна поліноміально еквівалентна складності задачі факторизації.

Розділ 3

Криптосистеми на еліптичних кривих

3.1 Криптосистемы на эллиптических кривых

(Автор: Семен Лигін. Російською мовою. Не редагувалось.)
(Версія від 18 січня 2017 р.)

На эллиптические системы переносятся только задачи, основанные на задаче дискретного логарифма

$$y = \alpha^x mod p$$

где α - примитивный элемент над F_p

Недостатки: зыбкость теоретического обоснования, медленная работа.

Первое понятно, по поводу второго: вот временна́я сложность.

$$T(n) = e^{cn^\nu (logn)^{1-\nu}}$$

$$0 <= \nu <= 1$$

$$\nu = 0 :$$

$$T(n) = e^{clogn} = n^c$$

$$\nu = 1 :$$

$$T(n) = e^{cn}$$

Если строго от 0 до 1 - то такие алгоритмы субэкспоненциальные Вообще ню примерно равно 1/3 В связи с этим создатели криптосистемы вынуждены увеличивать длину ключа, что не ускоряет работу системы. Сейчас длина ключи порядка 2048 бит.

В 1985 году Миллер и Коблитц предложили строить криптосистемы на эллиптических кривых. А именно: было предложено задавать на эллиптических кривых аддитивную абелеву группу. Собственно:

E - эллиптическая кривая, P - точка кривой большого порядка, под порядком имеем в виду порядок в группе.

$Q = xP = \underbrace{P + ... + P}_{\text{x}}$

И зная P, Q и E, затруднительно найти X.

Мультипликативная группа всегда циклична. А группа точек в E - ваще хз какая, любая может быть. Но это нормально.

Кстати, α - не обязательно должно быть примитивным элементом. Альфа должно иметь такой порядок, что невозможно создать таблицу для всех степеней α, примитивно говоря. В

конечном поле легко найти нужные элементы, вот их и берём. На практике берём кривые, порядки которых делятся на большое, просто n.

То есть, точки порядка эн.

и: $P, 2P, 3P, ..., nP = O_e$

- получается циклическая группа.

А это - задача дискретного логарифма (найти x по P,Q,E)

Задача дискретного логарифма на эллиптических кривых труднее, чем в конечном поле. Там мы работаем с простыми числами и неприводимыми полиномами, а на эллиптических кривых такого нет, следовательно, криптосистемы на эллиптических кривых при той же длинне ключа l более стойкие, чем в конечном поле.

В конечном поле ключи длины около 1024B, на ЭК - порядка 160B.

Но не всё так просто. В частности, схема Диффи-Хеллмана не переносится на ЭК тривиально, есть проблемы.

Вот есть алгоритм Диффи-Хеллмана:

A и B выбрали себе p и α - примитивный элемент F_p

$1 < K_A < p - 1$

$1 < K_B < p - 1$

$y_A = \alpha^{K_A} mod p -> y_A^{K_B} mod p = \alpha^{K_A * K_B} = K$

И в то же время B:

$y_B = \alpha^{K_B} mod p -> y_B^{K_A} mod p = \alpha^{K_A * K_B} = K$

И получают одинаковое K.

На эллиптических кривых он же:

$F_q, E, P : ordP = n$

$1 < K_A < n$

$1 < K_B < n$

$Q_A = \alpha^{K_A} * P -> y_{K_B} * Q_A = K_A * K_B * P = K$

$Q_B = \alpha^{K_B} * P -> y_{K_A} * Q_B = K_A * K_B * P = K$

Кажется, что они тривиально переносятся. Но есть проблемы.

КС на ЭК были предложены в 1985 году, но стандарты на них появились около 1999 года. Вот в схеме Диффи-Хеллмана: выбираем E, в которой есть точка большого порядка. А как выбрать-то?Нельзя просто посчитать руками.

Есть алгоритм Скуффа(Schoof)

$P \in Poly$, но если кривая над F_q, то $O((log 9)^9)$

Многовато.

Также известно неравенство Хассе для определения порядка эллиптической кривой.

Теорема 3.1 (Неравенство Хассе)**.** *Пусть N - порядок эллиптической кривой.*

E - кривая над F_q

Тогда

$$q + 1 - 2 * (q)^{-1/2} <= N_E <= q + 1 + 2 * (q)^{-1/2}$$

Доведення. Поле берём характеристики не два и не три. Подставляем X и смотрим на квадратичный вычет.

В среднем половина вычетов и половина невычетов.

Это $q/2$, в каждом есть по $2y$ - это единица, и $2q^{1/2}$ - это дисперсия.

□

Есть масса способов получить одну любую ЭК, например:

$y^2 = x^3 + ax + b$ над F_q, a,$b \in F_q$

Кривую рассмотрим над ${F_q}^n$ Правда, и кривая тогда имеет больше точек.

Рассмотрим такую кривую. В неравенстве Хассе числа в промежутке лежат очень узко (в каком-то смысле, мне непонятном)

$W F_Q {F_Q}^2 ... {F_Q}^r$
$N_1 N_2 ..... N_r$

Табл. 3.1: Такую табличку строим

Пусть N - число из интервала, заданного неравенством Хассе. Пусть N_1 - порядок кривой с A и B.

И есть итеративные формулы, чтобы считать N_i из N_j.

Нашли N_1, по нему считаем N_2 и проверяем, что N_2 делит P, где P - большое простое число.

Чуть-чуть подробнее об этой проверке. "При помощи кофактора, C - маленькое z^*"

Число делим на маленькие числа и затем проверяем на простоту. Да - ок. Нет - берём следующее число.

Перебирая находим кривую из F_q, и на ${F_q}^n$ будет тот же порядок.

Теперь ещё раз. Есть уравнение кривой. Рассмотрим её над F_q. А надо построить над ${F_q}^n$.

$$y^2 = x^3 + ax + b$$

$$F_q \to {F_q}^r$$

Знаем, что

$$q + 1 - 2 * (q)^{-1/2} <= N_E <= q + 1 + 2 * (q)^{-1/2}$$

И что:

$$N_1 \to N_2 \to N_3 \to ... \to N_r$$

W - первое число из ***. Оно соответствует какой-то кривой.

Считаем N_r

Проверяем, делится ли N_r на большое простое число.

Также следует понимать, что в маленьком поле мы можем просто непосредственно посчитать порядок кривой.

Схема шифрования Эль-Гамаля

Обычно:

$A : p, \alpha$

$1 < k < p - 1$ - секретный ключ

$y = \alpha^k modp$ - открытый ключ

$B : m < p$

$1 < X_m < p - 1$ - генерится каждый раз заново

$_1 = \alpha^{X_m} modp$

$_2 = y^{X_m} * mmodp$

$B : (C_1, C_2) -> A$

$A : (C_1^{-k} * C_2 = \alpha^{-k*X_M} * \alpha^{k*X_m} * mmodp = m$

На эллиптических кривых есть проблема: мы шифруем точку эллиптической кривой, а надо шифровать число. И нет такого детерменированного алгоритма, который сможет их сопоставить.

$$0 < m < M$$

Пусть - над полем характеристики q, не 2 и не 3.

M-1 - значение наибольшего сообщения.

$M - 1 < q$

$k : 1/2^k$ - вероятность не сопоставить точку.

Каждому M можем сопоставить отрезок. Берём точку и подставляем правую часть ($x^3 + ax + b$)

Это и будет точкой эллиптической кривой $m- > P_m$

Если квадратичный невычет - то берём следующее число в этом отрезке.

И тогда: пусть 1/2, что квадратичный вычет.

Тогда $1/2^k$ - вероятность не найти. Это ок.

$A : F_q, E, P$(большого порядка),$ordP = N$ $0 < k < n$ - k - секретный ключ. Q - открытый ключ, $Q = k * P$

$B : 0 < X_m < n, m- > P_m$, последнее - сопоставляет точку.

$C_1 = x * p$

$C_2 = X_m * Q + p_m$

$A : (-k) * C_1 + C_2$

$(-k) * p = k * (-p)$

- взаимная однозначность Например, 100p надо считать:

$100 = 2^6 + 2^5 + 2^2$

p,$2p$,2^2p,2^3p,2^4p,2^5p,2^6p

И складываем:

$2^2p + 2^5p + 2^6p = 100p$ - схема Горнера

Важно. Вот допустим есть у нас формулы сложения эллиптической кривой. Но ведь там же есть деление. Соответственно, кривую рассматривают в других координатах, например, в проективных.

А именно:

F_q ($\lambda * x$,$\lambda * y$,$\lambda * z$)

x, y, z и λ - элементы поля F_q

Между координатами в этих системах есть соотношение:

$(\lambda * x,\lambda * y,\lambda * z)- > (x/z, y/z, 1)$

И, если есть уравнение эллиптической кривой $y^2 = x^3 + ax + b$

То в проективных координатах:

$$y/z^2 = x/z^3 + ax/z + b$$

$$y^2 * z = x^3 + ax * z^2 + b * z^3$$

А это уравнение однородное.

Это позволяет выписать формулы для точек без деления.

А вообще есть и ещё другие координаты: смешанные - удвоение в одних, а сложение в других.

А ещё есть алгоритм Ито-Цудзии.

А ещё удобно делать некоторые операции в ОНБ.

3.2 Криптосистеми на еліптичних кривих

(Автор: Оксана Науменко. Не редагувалось.)
(Версія від 18 січня 2017 р.)

Перші стандарт криптосистем на еліптичних кривих:
1999 рік – ISO – стандарт на базі цифрового підпису Ель-Гамаля;
2000 рік – ECDSS – американський стандарт на базі звичайного підпису Ель-Гамаля;
2001 рік – ГОСТ Р34.10-2001 – російський стандарт цифрового підпису;
2002 рік – ДСТУ-4145-2002 – український стандарт, який відрізняється від російського та

американського (не являє собою кальку з російського стандарту)
Всі ці стандарти реалізують цифровий підпис Ель-Гамаля.
Існує множина різновидів Ель-Гамаля.
Рівняння підпису: $H = ks + rx \ mod(p-1)$, тут k - разовий секретний ключ (СК)
Рівняння перевірки підпису: $ak = bx + c \ mod(p-1)$, $a, b, c = \pm 1, \pm H, \pm r, \pm s, \pm H \cdot r, \pm H \cdot s, \pm r \cdot s, \pm ...$
Всі стандарти розглядаються над різними полями.
ISO – поле характеристики, більшої за 3, можуть бути як прості поля, так і розширені;
ГОСТ – тільки прості поля, що мають велику характеристику;
ДСТУ – поля характеристики 2, придумав - Кочубинський А.І.

ДСТУ-4145-2002

1. Вибір поля F_{2^m}., $163 \leq m \leq 509$
   163 та 509 – прості числа, і між ними зосереджено ще 60 простих чисел, тому для розглядання пропонуєть 60 полів характеристики 2.
   m – просте, тому що всі елементи мультиплікативної групи поля (крім одиниці) - примітивні.

   $F_{2^2}, F_{2^3}, F_{2^5}$ – p^{n-1} - просте, порядки максимальні.

   Серед 60 полів 14 мають нормальний оптимальний базис.

2. Вибір еліптичної кривої (ЕК).
   Еліптична крива обов'язково повинна бути несуперсингулярною!
   $y^2 + xy = x^3 + ax^2 + b$ (суперсингулярна має рівняння: $y^2 + y = x^3 + ax + b$)

   Операція ділення - дуже витратна операція, а для суперсингулярної кривої в рівнянні для пошуку кута нахилу немає ділення.
   MOV - атака:
   (P, 2P, 3P, ...) - циклічна група, що вкладена в мультиплікативну групу поля. (F_{2^m} вкладена в $F_{2^{km}}$)
   Для суперсингулярних кривих: $k \leq 6$ (тобто поле, в яке можна вкласти, не має бути більшим за $F_{2^{6m}}$)
   В задачі дискретного логарифма є метод аналізу таких полів(невеликих), тому криптосистеми, побудовані на суперсингулярних кривих, втрачають стійкість.

   Для несуперсингулярних кривих:
   (P, 2P, 3P, ... , nP=O_E), n не ділиться на $(2^{km} - 1)$, k - різні
   E=y^2+xy+x^3+ax^2+b, $b \neq 0$, a=0,1 (всі інші a будуть ізоморфні до a=0,1)
   $ordE$=cn, n - велике просте число (всі елементи групи будуть примітивними)
   В стандарті запропоновано 15 кривих. c= 2 або c= 4 (чим менше значення c, тим краще), задані a, b.
   Стійкість на *MOV-атаку* перевіряється до $k \leq 32$ (Для порівняння - в ISO перевіряється до $k \leq 20$, а це вже досить маленьке значення)
   На ЕК обирається точка $G \in E : ordG = n$ - базова точка (аналог α в ЦП Ель-Гамаля)

3. Вибір алгоритма хешування.
   За замовчуванням використовується український стандарт хешування ДСТУ–34.311 (ГОСТ Р-34.11)
   Хеш має 256 біт, $n > 2^{160}$
   Не забороняється використовувати інші хеші (при цьому додається ідентифікатор хешу).
   Абонент (А) обирає секретний ключ: d_A - випадкове число в проміжку $0< d_A <n$, СК є постійним і видається центром сертифікації.
   Відкритий ключ А - точка еліптичної кривої: $Q_A = -d_A \cdot G$ (тобто точку (-G) складають з собою d_A разів)
   Згадаємо: для несуперсингулярної кривої точка P має координати x та y, точка -P має координати x та $x + y$.

4. Процедура підпису.
   Маємо повідомлення M : $H(M)$ - число, що має довжину 256 біт (інтерпретується як елемент поля F_{2^m})
   Довжина хешу може не співпадати з довжиною елементів поля:
   –якщо $H(M)>h \in F_{2^m}$, відкидаємо старші біти;
   –якщо $H(M)<h \in F_{2^m}$, покладаємо старші біти нулями (00..0$H(M)$)
   M: $H(M) \to h \in F_{2^m}$, $h = 0 \Rightarrow h$=1 (якщо відкинули старші біти і залишились лише самі нулі)
   Абонент обирає разовий ключ: $0<k_A < n$.
   Абонент рахує $R = k_A \cdot G$ (DOPISAT"""")
   R має координати $(x_R, y_R), x_R \in F_{2^m}$, $y = x_R \cdot h \in F_{2^m}$ (множимо як елементи поля F_{2^m})
   Поліноми, що даються в полі F_{2^m} - триноми або пентаноми - незвідні.
   y переводимо в число $r<n$.
   Перша частина цифрового підпису : r.
   Рівняння підпису : $S = k_A + d_A \cdot r \ (mod n)$
   ДСТУ має особливість, яка підвищує його стійкість : h сховано мультиплікативним чином, а також перевагу в тому, що поля мають характеристику, рівну 2.

5. Сам підпис.

   DS =$(0||s||0||r)$ - нарощування s, r в старших бітах. – довжина повинна дорівнювати 16.
   – $0||s$, $0||r$ повинні бути рівними за довжиною. Маємо вектор : (iH, M, DS), i - індикатор.

6. Перевірка підпису.

   G помножене на число - аналог перевірки α^k в ЦП Ель-Гамаля.
   $S \cdot G = k_A \cdot G + d_A \cdot G \ \underbrace{k_A \cdot G} = S \cdot G - \underbrace{d_A \cdot G} \cdot r$, де $k_A \cdot G = R$, $d_A \cdot G$ - відкритий ключ $Q_A \Rightarrow R' = S \cdot G + Q_A \cdot r$ - можна порахувати (r - відомо).
   R' має координати $(x_{R'}, y_{R'})$. Той, хто перевіряє підпис, робить ті самі дії, що і абонент, який ставить цей підпис: $H(M) \in F_{2^m}, x_{R'} \cdot h = y \in F_{2^m}$ x_R - *передпідпис.*
   Дозволяється генерувати багато передпідписей, але потім знищувати їх.

3.3 Цифрова готівка

Особливість цифрової готівки - анонімність платника. ЦГ базується на алгоритмі сліпого цифрового підпису (на базі RSA). *Сліпий ЦП.* Маємо A - абонент, що вимагає цифрового підпису від B та схему RSA з n, e, d
A: хоче, щоб B поставив підпис, але не зміг прочитати повідомлення.
$\underbrace{M \cdot r^e} \to B$, r - випадкове число, таке, що $(r,\ n)$=1, $M \cdot r^e = M_1, r$ - осліплюючий множник (множник, що затемнює, засліплює)
B: (підписує M_1)$M_1^d = Smodn \longrightarrow A$
A: $M_1^d = M^d \cdot r^{ed} modn = M^d modn = S_1$
$S_1 \cdot r^{-1} = M^d \cdot r^{-1} \cdot r = M^d modn$
B - той, хто підписує, і він не бачить повідомлення M.
Схема цифрової готівки.
Маємо схему, що складається з трьох елементів: T - торговець, $Б$- банк, K- клієнт. K і T відкривають рахунок в $Б$ (повинні покласти деяку суму грошей).
K створює електронну купюру: *(v, n)*, де v - *value* - номінал (наприклад, 100 доларів), n - дуже велике випадкове число (номер купюри).
K затемнює *(v, n)* і відправляє до $Б$, повідомляючи при цьому значення v.
$Б$ підписує сліпою ЦП, не знаю при цьому n купюри; $Б$ знімає з рахунка K купюру зазначеного номіналу v.

Виникає питання: чому банк довіряє клієнтові? Адже він може зазначити будь-яку суму, в рази меншу, ніж потрібну. Для того, щоб банк повірив клієнтові, виконуються такі дії:
K формує m електронних купюр одного номіналу v так, щоб ймовірність $\frac{1}{m}$ була малою : $(v, n_1), (v, n_2), ..., (v, n_m)$ і надсилає $Б$.
$Б$ випадковим чином обирає одну з купюр $1 \leq j \leq m$ і просить надіслати сліпучий множник на всі, крім обраної j-тої купюри.
Таким чином $Б$ перевіряє: якщо всі номінали розкритих купюр v співпадають, то $Б$ ставить свій підпис.
Потім K надсилає електронну купюру T : *(v, n, S)* - гроші (клієнт знімає сліпучий множник).
T перевіряє підпис банку та надсилає свою електронну купюру до $Б \to Б$ має $(v, n), n_K = n_T$.
Ще один варіант шахрайства: K і T можуть надіслати купюру з однаковим n. Для цього існує реєстр використованих купюр.

Розділ 4

Асиметричні криптографічні протоколи

4.1 Теорія імітостійкості

(Автор: Іван Свічкарьов. Не редагувалось.)
(Версія від 19 січня 2017 р.)

Ми вже познайомилися з таким центральним поняттям асиметричною криптографії, як цифровий підпис, який підтверджує цілісність повідомлення та справжність джерела.

Виникає питання: а чому в симетричній криптографії, якщо відбувається шифрування якогось повідомлення і при розшифровці законним користувачем ми отримує осмислений текст, то чому при цьому не забезпечується автентичність повідомлення?

Виявляється, ці питання вирішуються неоднозначно. Раніше вважалося, що якщо зашифрували ВТ і відправили законному користувачеві і він розшифрував його, то воно є цілісним. Але це виявляється не зовсім так. І цю теорію в 80-х роках розвинув Сіммонс, схожу на теорію Шеннона. Розглянемо її більш детально.

Питання автентичності та цілісності повинні забезпечуватися не просто секретним ключем, а й якимись додатковими алгоритмами, додатковими ключами.

Розглянемо систему симетричного шифрування. Загальна схема яку зараз буде розглядатися схожа на схему Шеннона, але тут криптоаналітик вже в більш вигідному положенні. Тобто він вже може не тільки знімати ШТ, а й втручатися у передачу.

На рисунку 4.1: M, C – це власне відкритий і шифрований текст; $\widetilde{M}, \widetilde{C}$ – якісь змінені ВТ і ШТ відповідно, якщо криптоаналітик E вніс якісь зміни. Зауважимо, що в схемі Шеннона $M = \widetilde{M}$ і $C = \widetilde{C}$. Ну і звісно K є закритим ключем.

Пунктирна лінія означає, що передача може бути припинена і передана інформація - змінена.

Математична модель криптосистеми

З математичної точки зору, криптосистема є сукупністю просторів

$$\Sigma = \langle \mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle ,$$

де $\mathcal{M}$ – простір відкритих текстів,
$\mathcal{C}$ – простір криптограм,
$\mathcal{K}$ – простір ключів,
$\mathcal{E}$ – простір алгоритмів шифрування,
$\mathcal{D}$ – простір алгоритмів розшифрування.

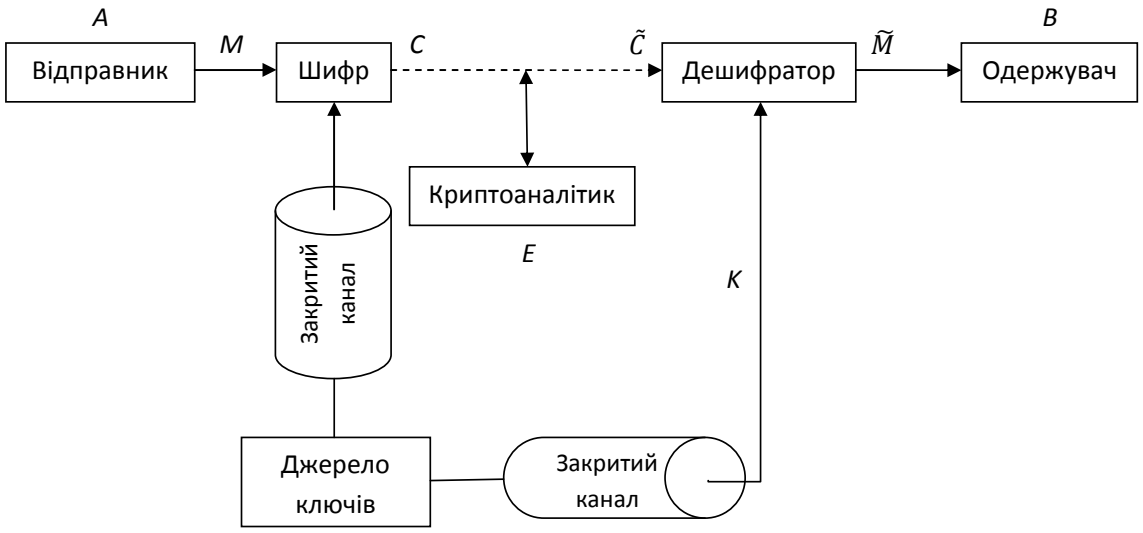


Рис. 4.1: Загальна схема секретного зв'язку

Далі будемо вважати, що $E_k \in \mathcal{E}$ и $D_k \in \mathcal{D}$ – конкретні перетворення, тобто алгоритми шифрування і розшифрування на ключі k відповідно.

У загальному випадку під *шифром* визначається наступне:

$$\text{це відображення } \mathcal{M} \times \mathcal{K} \to \mathcal{C} : \forall k \in \mathcal{K}, M \in \mathcal{M} : D_k(E_k(M)) = M,$$

де $E_k : \mathcal{M} \to \mathcal{C}$ – ін'єктивне відображення, яке дає можливість зашифрувати будь-який ВТ на ключі k, ну і відповідно при цьому, також відбувається однозначне розшифрування.

Основні припущення Сіммонса

1. Атака криптоаналітика E здійснюється за допомогою ШТ.

Криптоаналітик E хотів би обдурити B, пославши якусь іншу криптограму і щоб він подумав, що це йому відправив A. Якщо така дія вдається E, то вважається, що абонент B введений в оману.

2. На декартовому добутку $\mathcal{M} \times \mathcal{K}$ задано ймовірнісний розподіл.

Тобто, така ймовірність $P(M,K) : \forall M \in \mathcal{M}, K \in \mathcal{K}$ виконується наступне:

$$\sum_{M,K} P(M,K) = 1,\ \sum_{M} P(M,K) = P(K),\ \sum_{K} P(M,K) = P(M).$$

Згідно з правилом Керкгоффса, всі простори криптосистеми Σ і розподіл на $\mathcal{M} \times \mathcal{K}$ вважаються відомими криптоаналітику.

Подивимося, як же Сіммонс розвивав свою теорію далі.

Він розглянув 2 типи атак:

1. **Імітація**: Криптоаналітик E не очікує справжній ШТ від A, тобто не перехоплюючи, формує помилковий ШТ $\widetilde{C}$ і відправляє його B.

Атака з імітацією вважається успішною, якщо B вважає, що $\widetilde{C}$ допустима криптограма від A, тобто $\widetilde{M}$ – ВТ, прийнятий за допустимий текст A, навіть якщо B пізніше відправив ШТ $C = \widetilde{C}$. Тобто ця криптограма не розпізнана і в цьому випадку, якщо B був введений в оману, то незалежно від того який там був текст – помилковий чи правильний, все одно атака є успішною через те, що криптограма прийшла не в той час, коли відправляв A і відповідно висновки з отриманої криптограми будуть інші. Тому що важливо не тільки зміст криптограми, а й коли вона прийшла.

Позначимо ймовірність імітації :

$$P_{\text{ім}} = \max_{\widetilde{C} \in \mathcal{C}} P(\widetilde{C} - \text{допустима}) \tag{4.1}$$

Тобто, щоб обдурити B, криптоаналітик E намагається вибрати таку криптограму $\widetilde{C}$, щоб максимізувати ймовірність (4.1).

2. **Атака підміни**: E перехоплює справжній ШТ C від A і формує помилковий ШТ $\widetilde{C}$ і передає його B.

Атака вважається успішною, якщо B прийняв $\widetilde{C}$ за допустиму.

Позначимо ймовірність підміни:

$$P_{\text{підм}} = \max_{\substack{\widetilde{C} \neq C \\ \widetilde{C} \in \mathcal{C}}} P(\widetilde{C} - \text{допустима})$$

Тоді ймовірність обману користувача B буде визначатися так:

$$P_{\text{об}} = \max\{P_{\text{ім}}, P_{\text{підм}}\}$$

Як визначити систему, яка буде цілком таємною, тобто найкраща з точки зору криптостійкості. То тут цілком природньо було б визначити абсолютно автентичну криптосистему. Але як її визначити, тобто найкращу з точки зору неможливості підробки, обману або такої імітації, не зовсім зрозуміло. І це вже не зовсім тривіальне питання, тому що, якщо передається якесь повідомлення, то така ймовірність обману все одно буде, як ми незабором це побачимо.

Проілюструємо загальну схему роботи криптосистеми на малюнку (4.2).
Розглянемо скінченну множину відкритих текстів M, яке зазвичай є певною підмножиною більшої множини і множину шифрованих текстів C, яке зазвичай більше за множину ВТ. Також є множина ключів, яка практично збігається з усім простором ключів, за виключення нетипових слабких ключів.

Нехай обрані ключі K_1, K_2, K_3, на яких працюють абоненти. То ми можемо сказати, що весь простір ВТ відобразиться в деякий підпростір ШТ і позначимо їх C_1, C_2, C_3 відповідно, які в свою чергу можуть перетинатися, а можуть і не перетинатися.

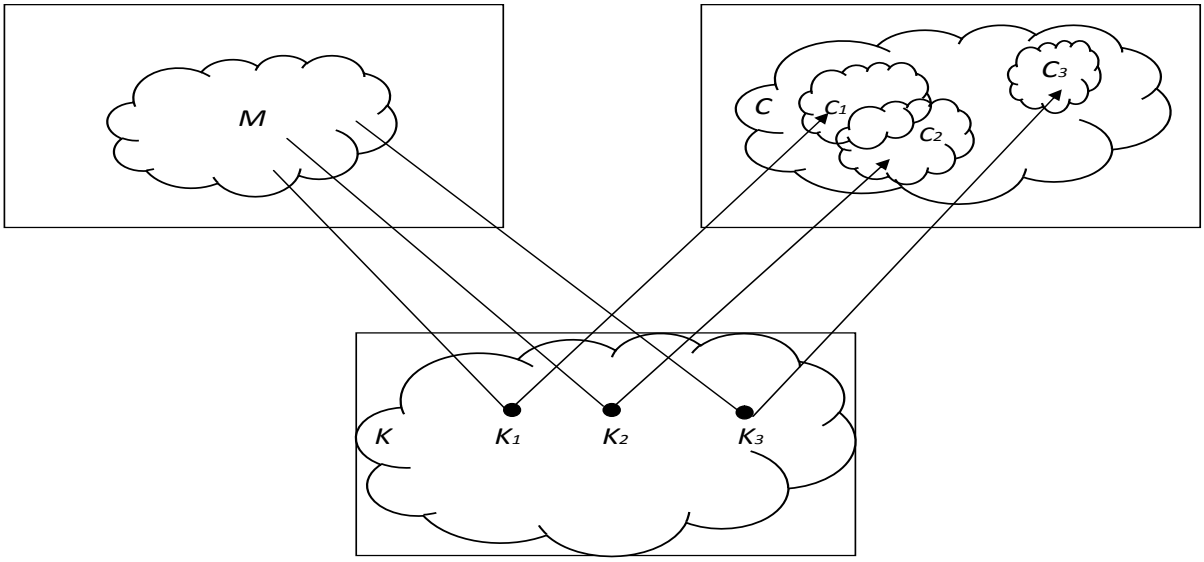


Рис. 4.2: Ілюстрація загальної схеми секретного зв'язку

Розглянемо атаку імітації.

Нехай A і B працюють на ключі k і відповідно заданий алгоритм шифрування на цьому ключі: $E_k(\mathcal{M}) = \mathcal{C}_k$, де $\mathcal{C}_k \in \mathcal{C}, k \in \mathcal{K}$.

Тоді, якщо абонентами A і B вибирається випадкове k, а криптоаналітиком E таке $\widetilde{C} \subset \mathcal{C}$ і відправляє $\widetilde{C} \to B$, то ймовірність успіху атаки:

$$P(\widetilde{C} - \text{допустима}) = \frac{|\mathcal{C}_k|}{|\mathcal{C}|} \geq \frac{|\mathcal{M}|}{|\mathcal{C}|} > 0, \tag{4.2}$$

причому $|\mathcal{C}_k| \geq |\mathcal{M}|$ через однозначність розшифрування. Також ясно, що $|\mathcal{M}| > 0$. Ну і власне ймовірність (4.2) більше нуля, з чого і випливає, що така ймовірність обману існує.

В нерівності (4.2) досягається рівність $\Leftrightarrow \forall k \in \mathcal{K} : |\mathcal{C}_k| = |\mathcal{M}|$,
тобто $\forall k \in \mathcal{K} : E_k : \mathcal{M} \to \mathcal{C}_k$ - бієкція.

Виникає питання, а як же зробити ймовірність обману меншою?

Коли застосовуємо до повідомлення цифровий підпис, то фактично ймовірність обману зменшується через те, що збільшується простір ШТ. Далі буде з'ясовано, яким чином вирішити це питання за допомогою імітовставок.

У загальному випадку K вибирається з ймовірністю $P(K)$.

Позначимо через:

$$\mathbb{1}_K(C) = \begin{cases} 1, & \text{якщо } C \in \mathcal{C}_K \\ 0, & \text{якщо } C \notin \mathcal{C}_K \end{cases} - \text{індикатор } C,$$

тобто $\mathbb{1}_K(C) : \mathcal{C} \to \{0, 1\}$

Ймовірність того, що $\widetilde{C}$ - допустима, визначається наступним:

$$P(\widetilde{C} - \text{допустима}) = \sum_{K \in \mathcal{K}} P(K) \cdot \mathbb{1}_K(\widetilde{C}) \tag{4.3}$$

Ми не знаємо секретного ключа K, але ми знаємо його розподіл, через що у нас і буде визначена ймовірність (4.3). Тоді:

$$P_{\text{ім}} = \max_{\widetilde{C}\in\mathcal{C}} P(\widetilde{C} - \text{допустима}) = \max_{\widetilde{C}\in\mathcal{C}} \sum_{K\in\mathcal{K}} P(K)\cdot \mathbb{1}_K(\widetilde{C}) \tag{4.4}$$

Шляхом деяких технічних міркувань, з використання нерівності Йєнсена, Сіммонс оцінив ймовірність (4.4):

$$P_{\text{ім}} \geq 2^{-I(\mathcal{C},\mathcal{K})}\,, \tag{4.5}$$

де $I(\mathcal{C},\mathcal{K}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{K})$ – взаємна інформація.

При цьому, в нерівності (4.5) рівність досягається $\Leftrightarrow$ виконуються наступні умови:

1. $P(\widetilde{C} - \text{допустима})$ не залежить від $\widetilde{C}$. Тобто оптимальна атака – це випадковий вибір криптограми $\widetilde{C}$.

2. $P(\widetilde{C}|K)$ однакова для $\forall K \in \mathcal{K} : \widetilde{C} \in \mathcal{C}_K$, тобто $\mathbb{1}_K(\widetilde{C}) = 1$.

Тоді ймовірність обману визначається наступним:

$$P_{\text{об}} = \max\{P_{\text{ім}}, P_{\text{підм}}\} \geq 2^{-I(\mathcal{C},\mathcal{K})}\,. \tag{4.6}$$

Визначення 4.1. Абсолютно автентичною криптосистемою називається така, для якої в нерівності (4.6) досягається рівність.

Ймовірність (4.6) не може бути нульовою, але вона може бути мінімізована, коли взаємна інформація між $\mathcal{C}$ і $\mathcal{K}$ в свою чергу максимізована. Трактується це наступним чином: наскільки ключ використовується для створення умов імітостійкості.

Приклад 4.1. $\mathcal{M} = \{M_1, M_2\}$, $\mathcal{K} = \{K_1, K_2\}$, де кожна подія приймається з однаковою ймовірністю. $\mathcal{C} = \{00, 01, 10, 11\}$.

Задамо алгоритм шифрування у вигляді таблиці (4.1), при якому показано, якому ШТ відноситься відповідний ВТ на заданому ключі. Також ми подивимось, чи можна зробити ймовірність імітації рівній $\frac{1}{2}$ і яка ж буде ймовірність обману.

K \ M	M_1	M_2
K_1	00	10
K_2	01	11

Табл. 4.1: Алгоритм шифрування

Однозначність розшифрування в даному прикладі забезпечена, так як видно, що при кожному ключі - ВТ переходить в якийсь відмінний від інших ШТ.

Знайдемо взаємну інформацію між $\mathcal{C}$ і $\mathcal{K}$:

$$I(\mathcal{C},\mathcal{K}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{K}) = \log(4) - \log(2) = 1$$

Якщо криптоаналітик посилає якусь довільну криптограму і якщо A і B працюють на ключі K_1, то він завжди обдурить, якщо вибере один з двох ШТ: 00 або 10. Тобто він з ймовірністю $\frac{1}{2}$ зможе здійснити атаку імітації і таким чином в нерівності (4.5) досягається рівність.

Але чому ж дорівнює ймовірність підміни?

Можна помітити, що якщо ключ K_1, то в криптограмі останній біт – 0, на ключі K_2, відповідно – 1. Таким чином, ми по криптограмі можемо зрозуміти на якому ключі працюють A і B, і фактично ця система з точки зору стійкості – погана. Тоді ймовірність підміни буде дорівнює одиниці, через те якщо ми перехопимо криптограму 00, то зразу посилаємо 10 і навпаки. Ну і звичайно ж ймовірність обману строго більше $\frac{1}{2}$, тобто в (4.6) рівність не досягається.

З усього цього слідує те, що наш шифр не є абсолютно автентичним. Ну і звичайно ж зрозуміло, щоб зменшити ймовірність підміни, необхідно розширювати простір ШТ. Для цих цілей використовують імітовставки.

Імітовставки

Нехай E_K – алгоритм блочного шифру, а повідомлення M розбивається на блоки, тобто $M = M_1 M_2 \dots M_t$ і в режимі зчеплення блоків при таємному ключі K знаходимо

$$H_i = E_K(M_i \oplus H_{i-1}) \, , i = \overline{1,t}$$

Розмір блоку дорівнює розміру входу і виходу. тобто $|M_i| = |E_K(X)| = |X|$, відповідно H_0 - вектор ініціалізації, а H_t - імітовставка.

Повідомлення з імітовскавкой (M, H_t) забезпечує автентичність з використанням секретного ключа, і при цьому простір отриманих текстів розширюється і далі воно може бути зашифровано іншим ключем. Для цієї мети також використовуються геш-функції із секретним ключем – $h(X,K)$.

Властивості $h(X,K)$:

1) Швидке обчислення.

2) $\forall M \in \mathcal{M}$ не можна вгадати значення геш-функції $h(M,K)$ з ймовірністю більш ніж $2^{\min\{m,|K|\}}$, де $|K|$ – довжина ключа в бітах, m – розмір виходу $h(X,K)$.

3) Для скінченного набору: $\{M_i, h(M_i,k)\}_{i=1}^t$ не можливо знайти хоча б одну пару $(M, \widetilde{M}) : H(M,k) = h(\widetilde{M},k), M \neq \widetilde{M}$ за поліноміальний час.

Приклад геш-функцій з ключем: імітовставки.

4.2 Ідентифікація та аутентифікація. Криптографічні протоколи аутентифікації

(Автор: Марина Соловйова. Не редагувалось.)
(Версія від 19 січня 2017 р.)

Ідентифікація - це надання суб'єкту (об'єкту) індивідуального кода (наприклад, номера), тобто ідентифікатора, який відрізняється від всіх інших, а також перевірка ідентифікатора(все це не секретні дані).

Аутентифікація - це перевірка достовірності суб'єкту по наданим ідентифікаторам та деякій додатковій інформації (наприклад, паролям, ключам) шляхом порівняння.

При контакті користувача, наприклад, з комп'ютерною мережею здійснюються наступні процедури:

– ідентифікація

– аутентифікація

– авторизація

Принципи аутентифікації

1. По наданню деякої секретної інформації, відомої користувачеві (пароль, код).

2. По наданню деяких об'єктивних характеристик (мікросхема, магнітна стрічка).

3. По наданню індивідуальних біометричних характеристик (відбитків пальців, рисунку райдужної оболонки ока, тембру голосу).

4. По характеристикам роботи з апаратурою в реальному часі.

5. По відповідям в режимі діалогу.

6. За допомогою третьої сторони.

Криптографічні протоколи аутентифікації

1. Аутентифікація за допомогою пароля (на прикладі аутентифікації з комп'ютерною системою - надалі будемо позначати як "КС").

   (a) *Парольна аутентифікація з використанням односторонніх функцій*
   Нехай h(x) - одностороння хеш-функція, що володіє сильною стійкістю до колізій.
   Користувачі A_i мають ідентифікатори D_i та паролі p_i, i = $\overline{1,n}$.
   1) Попередньо в КС формується та розміщується табличка
   T = $\{D_i, H_i, i = \overline{1,n}\}$, де $H_i = h(p_i,D_i)$.
   2) (При контакті) $A_i : (p_i,D_i) \to$ КС.
   3) КС перевіряє: $?D_i \in T?$ (ідентифікація),
   $?h(p_i, D_i) = H_i? \longleftrightarrow ?H_i \in T?$ (аутентифікація).
   Якщо все виконується, то авторизація, інакше - відмова.

   (b) *Паралельна аутентифікація за допомогою симетричного шифрування стійких до атак на основі відкритого тексту (ВТ).*
   Користувачі A_i мають ідентифікатори D_i та паролі p_i, $i = \overline{1,n}$.
   Нехай E_k - алгоритм симетричного шифрування, стійкий до атак на основі ВТ.
   1) Попередньо в КС формується та розміщується табличка
   T = $\{D_i, C_i, i = \overline{1,n}\}$, де $C_i = E_{p_i}(D_i)$.
   2) $A_i : (p_i,D_i) \to$ КС.
   3) КС перевіряє: $?D_i \in T?$ та $?E_{p_i}(D_i) = C_i?$
   Якщо все виконується, то авторизація, інакше - відмова.

   (c) *Парольна аутентифікація з захистом від коротких паролів.*
   - модифікація в варіанті 1 з хеш-функцією h(x) - використання односторонніх хеш-функцій з секретним ключем (який відомий лише адмін-у КС).
   - модифікація в варіанті 2: табличка T = $\{D_i, C_i, i = \overline{1,n}\}$ - використання секретного ключа k: $C_i = E_{p_i \oplus k}(D_i)$.
   Перевірка аналогічна.

   (d) *Парольна аутентифікація зі змінним паролем.*
   Користувачі A_i мають ідентифікатори D_i та паролі $p_i^{(0)}, i = \overline{1,n}$.
   Наявна одностороння стійка до колізій хеш-функція h(x) (відкрита).
   1) Кожен A_i обчислює ряд разових паролів: $p_i^{(0)}$, $p_i^{(1)} = h(p_i^{(0)})$, $p_i^{(2)} = h(p_i^{(1)}) = h^2(p_i^{(0)})$,..., $p_i^{(t)} = h^t(p_i^{(0)})$ та $p_i^{(t)} \to$ КС, $i = \overline{1,n}$.
   2) КС формує табличку T = $\{D_i, p_i^{(t)}, i = \overline{1,n}\}$.
   3) (При контакті) $A_i : (D_i, p_i^{(t-1)}) \to$ КС.
   4) КС перевіряє: $?D_i \in T?$ та $?h(p_i^{(t-1)}) = p_i^{(t)}?$
   Якщо все в виконується, то виконується авторизація і КС в табличці T змінює $(D_i, p_i^{(t)})$ на $(D_i, p_i^{(t-1)})$.

5) (При другому контакті) $A_i : (D_i, p_i^{(t-2)}) \to$ КС.
6) КС перевіряє: $?D_i \in T?$ та $?h(p_i^{(t-2)}) = p_i^{(t-1)}?$
Якщо все виконується, то КС в табличці Т змінює $(D_i, p_i^{(t-1)})$ на $(D_i, p_i^{(t-2)})$.
Примітка: Якщо всі разові паролі будуть вичерпані, потрібно згенерувати нові їх послідовності.
Але що робити при технічному збої? Зрозуміло, що повторювати пароль не можна. Тому при, наприклад, однократному збої на кроці 6): $A_i : (D_i, p_i^{(t-3)}) \to$ КС. Після чого КС перевіряє: $?D_i \in T?$ та $?h^2(p_i^{(t-3)}) = p_i^{(t-1)}?$ після чого алгоритм продовжує свою роботу.

2. Аутентифікація за допомогою симетричних криптосистем.

   Користувачі A_i, $i = \overline{1,n}$ мають ідентифікатори D_i та K_{ij} - секретні ключі A_i та A_j зі спільною системою симетричного шифрування (E_k, D_k).
   1) Ініціатор аутентифікації $A_i : D_i \to A_j$.
   2) A_j генерує випадкове $M \to A_i$.
   3) A_i шифрує на спільному ключі повідомлення: $E_{K_{ij}}(M) = C_{ij} \to A_j$.
   4) A_j перевіряє: $D_{K_{ij}}(C_{ij}) = M$.

3. Аутентифікація за допомогою асиметричних криптосистем (на прикладі RSA).

   Користувачі A_i, $i = \overline{1,n}$ мають відкриті (n_i, e_i) та закриті d_i ключі RSA.

   (a) За допомогою цифрового підпису (ЦП).
   1) A_i генерує випадкове М і підписує його, тобто:
   $S = M^{d_i} mod\ n_i, \Rightarrow (M, S) \to A_j$.
   2) A_j перевіряє ЦП: $?S^{e_i} mod = M?$

   (b) За допомогою шифрування.

4. Аутентифікація за допомогою протоколу доказу без розголошення.

4.3 Протоколи доведення з нульовим пізнанням(zero - knowledge protocol).

(Автор: Юля Старкова. Редагувалось(Грубіян))
(Версія від 21 січня 2017 р.)

Протоколи з нульовим пізнанням є одним з основних способів безпечної автентифікації користувачів.
Сторони протоколу ДНП:
A - Сторона що доводить
B - Сторона що перевіряє
Ціль(задача) протоколу ДНП(не прив'язуючись до певного алгоритму):
A має секрет та доводить **B**, що знає секрет, але так, що В про секрет, в результаті протоколу, не дізнається ніякої інформації
Подивимось на конкретні протоколи:

4.3.1 Протокол ДнП на основі квадратичних лишків

A має число $n = pq$, $p \neq q$ великі прості p і q - секрет **A**, а n „індентифікатор“**A**
Ціль протоколу ДнП: **A** доводить **B**, що знає p та q , при цьому B ніякої інформації про p і q не отримує.

КРОКИ ПРОТОКОЛУ

1. **B**: генерує випадковий $x, (x,n) = 1$; обраховує $x^4 \bmod n = y \longrightarrow$ **A**, з вимогою знайти квадратний корінь з y, який є квадратичним лишком.

2. **A**: Знаходить корінь $y^{1/2} \bmod n = \{z_1, z_2, z_3, z_4\}$, знаючи та використовуючи p і q. З чотирьох коренів, він обирає квадратичний лишок (знаючи p і q згідно з критерієм Ойлера, обраховуємо символ Лежандра) $z^* \longrightarrow$ **B**

3. **B** перевіряє: $z^* = x^2 \bmod n$. Якщо „так“, то **B** доведено, що **A** знає p та q
   Подивимось, яку інформацію отримає **B**: він отримав z^*, але оскільки x і $x^2 \bmod n$ **B** знав раніше, то про p і q **B** ніякої інформації не отримує.

Атака на протокол

1. **B**: генерує випадковий x, $(x,n) = 1$.
   Обраховує $x^2 \bmod n = y$ (порушення протоколу)

2. **A**: знаходить $y^{1/2} \bmod n = \{z_1, z_2, z_3, z_4\}$ та квадратичний лишок $z^* \longrightarrow$ **B**.

3. **B**: перевіряє $z^* = x^2 \bmod n$. Випадки:

   a) $z^* = x \Rightarrow$ **A** доводить, що знає p і q , але **B** ніякої інформації про p і q не отримує.

   b) $z^* \neq x, z^* = -x \Rightarrow$ **A** доводить **B**, що знає p і q, але **B** інформації про p і q не отримує.

   c) $z^* = x, z^* \neq -x$ (**A**i); Тоді **B** має два кореня з y такі, що $z^* \neq \pm x \bmod n$ що $gcd(z^* + x, n) = p$ або q.
   ! Ймовірність випадку с (якщо число n - число Блюма) дорівнює $1/2$.
   Цей протокол не використовується, оскільки його секркткість базується на „порядності“**B**.

4.3.2 Протокол ДнП на базі квадратичності

Задача: **A** знає пару (x, y) таку, що $y = x^2 \bmod n$, $n = pq$, $p \neq q$ - великі прості. При цьому **A** не знає p і q. **A** „оголошує“y, n та доводить **B**, що знає x - один з коренів $y^{1/2} \bmod n$ таким чином, що ніякої інформації про x **B** не отримує.

КРОКИ ПРОТОКОЛУ

1. **A**: генерує випадкове $v, (v,n) = 1$, лишок $u = v^2 \bmod n, n \longrightarrow$ **B**

2. **B** генерує випадковий біт.

$$b = \begin{cases} 1, & \text{з ймовірністю } 1/2 \\ 0, & \text{з ймовірністю } 1/2 \end{cases}$$

$$b \longrightarrow \mathbf{A}$$

3. **A**: Якщо $b = 1$, то $v \longrightarrow \mathbf{B}$; Якщо $b = 0$, то $vx = w \longrightarrow \mathbf{B}$

4. **B**: Якщо $b = 1$, то перевіряємо $v^2 = u \bmod n$; Якщо $b = 0$, то перевіряємо $w^2 = uy \bmod n$ Кроки 1 - 4 перевіряються m разів $m < \log n$. Якщо на всіх m ітераціях відповіді „так“, тоді **A** довів **B** , що знає такий x, що $x^2 \bmod n = y$.Імовірність обману $\leqslant 2^{-m}$. Якщо, хоча б на одній ітерації отимуємо відповідь „ні“, тоді **A** не довів **B**, що знає x.

   Розглянемо можливість обману, коли **A** не знає x, $x^2 = y \bmod n$(ймовірність обману на кожній ітерації - $1/2$)

   a) $U \longrightarrow \mathbf{B}, U = V^2 \bmod n$. Тоді з імовірністю $1/2$ обман буде виявлено при $b = 1$.

   b) $U \longrightarrow \mathbf{B}, U = V^2 \bmod n$, y - не квадратичний лишок, тоді з ймовірністю $1/2$ при $b = 0$ обман буде виявлено, оскільки $w^2 \neq uy \bmod n$.

   c) $U \longrightarrow \mathbf{B}, U = V^2 \bmod n$. y - квадратичний лишок, тоді з ймовірністю $1/2$ при $b = 0$ обман буде виявлено на 4-му кроці, оскільки $w^2 \neq uy \bmod n$.

4.3.3 Протокол підкидання монети по телефону(схема Блюма-Мікалі)

Сторони протоколу:
A та **B** спілкуються по каналу зв‘язку
Задача: генерувати випадковий біт таким чином, щоб ні **A** ні **B** не вплинули на результат. Випадковим біт:

$$\gamma = \begin{cases} 1, & \text{з ймовірністю } 1/2 \\ 0, & \text{з ймовірністю } 1/2 \end{cases}$$

КРОКИ ПРОТОКОЛУ

1. **A**: Обирає випадковий k - ключ. Генерує випадкове $\alpha \in \{0,1\}$, шифрує, та надсилає $E_k(\alpha) \longrightarrow \mathbf{B}$.

2. **B**: генерує випадковий біт $b \in \{0,1\}$, $b \longrightarrow \mathbf{A}$

3. **A** : $(\alpha, k) \longrightarrow \mathbf{B}$.

4. **B**: перевіряє $\alpha = D_k(E_k(\alpha))$.

5. **A** і **B**: спільний секрет: $\gamma = \alpha \oplus b$.

Генерація ключів за допомогою підкидання монети

Реальним застосуванням цього протоколу є генерація сеансового ключа. Протокол підкидання монети дозволяє **A** і **B** створити випадковий сеансовий ключ таким чином, що ніхто з них не має можливості влинути на те, яким він буде.

4.3.4 Протоколи на основі RSA

Нехай **A** має RSA з відкритим ключем (n, e) секретний ключ d.

КРОКИ ПРОТОКОЛУ

1. **A**: Генеруємо випадковий $x \in \{1, 2, \ldots, n-1\}$ (однакова кількість парних та непарних). Обраховуємо $x \bmod 2 = \alpha \in \{0{,}1\}$. $x^e \bmod n = y \longrightarrow \mathbf{B}$.

2. **B**: Обираємо випадковий біт:

$$\beta = \begin{cases} 1, & \text{з ймовірністю } 1/2 \\ 0, & \text{з ймовірністю } 1/2 \end{cases}$$

3. **A** : $(n, e), x, \alpha \longrightarrow \mathbf{B}$.

4. **B**: Перевіряє (n, e) - відкритий ключ **A**(надсилає зашифроване повідомлення або перевіряє у центрі сертифікації, наприклад).

$$x^e \bmod n = y, \alpha = x \bmod 2$$

5. **A** і **B**: Якщо результат перевірки задовільняє сторони протоколу, то $\gamma = \alpha \oplus \beta$, α - випадковий біт.

$$\gamma = \begin{cases} 1, & \text{з імовірністю } 1/2 \\ 0, & \text{з імовірністю } 1/2 \end{cases}$$

4.4 Протоколи розділення секретів

(Автор: Олеся Столова. Не редагувалось)
(Версія від 20 січня 2017 р.)

A_i, $i = \overline{1,n}$-учасники протоколу, A_p-арбітр

Задача 1. A_p має деякий секрет S і цей секрет S "розділяє"між учасниками A_i, $i = \overline{1,n}$ так, що кожен з них не знає ніякої інформації про секрет S, будь-які $k < n$ учасників про секрет S також ніякої інформації не можуть отримати, а усі учасники A_i, $i = \overline{1,n}$ разом, повністю відновлюють S.

Задача 2. A_p секрет S "розділяє"між учасниками A_i, $i = \overline{1,n}$ так, що будь-які $1 \leq k < n$, секрет S відновлюють повністю, а будь-які $r < k$ учасників про секрет S ніякої інформації не отримають.

Вирішення цих задач криптографічними методами:

Протокол розділення секретів 1

Задача 1. Арбітр A_p має секрет $S = S_1,S_2,\ldots,S_t$, $S_i \in \{0{,}1\}$- випадкова послідовність. Розділення секретів арбітром A_p

1. Генерується $n-1$ випадкова послідовніть з незалежними та рівноймовірними бітами, які не залежать також від $S = S_1,S_2,\ldots,S_t$.

2. Арбітр учаснику A_i передає послідовність $S^{(i)}$, $i = \overline{1,n-1}$, а учаснику A_n передає послідовність $S^{(n)} = S \oplus \oplus_{i=1}^{n} S^{(i)}$.

Відновлення секрету S Учасники A_i, $i = \overline{1,n}$ разом відновлюють секрет по формулі $s = \oplus_{i=1}^{n} S^{(i)}$. Дійсно, $\oplus_{i=1}^{n} S^{(i)} = \oplus_{i=1}^{n-1} S^{(i)} \oplus S^{(n)} = \oplus_{i=1}^{n-1} S^{(i)} \oplus S \oplus \sum_{i=1}^{n-1} S^{(i)} = S$.

Чому про S менш ніж n учасників нічого не дізнається?

а) Будь-які учасники з перших $n-1$ вочевидь про S нічого не дізнаються.

б) Нехай в групі $r < n$ учасників присутні A^i, $i = \overline{1,r}$, $r < n-1$.

Нехай учасники складуть послідовність: $\oplus_{i=1}^{r} S^{(i)} \oplus S^{(n)} = S \oplus \sum_{i=1}^{n-1} S^{(i)} \oplus \sum_{i=1}^{r} S^{(i)} = S \oplus \oplus_{i=r+1}^{n-1} S^{(i)}$ (1) (Аналогія шифру Вернона) звідси дізнатися про S якусь інформацію-неможливо.

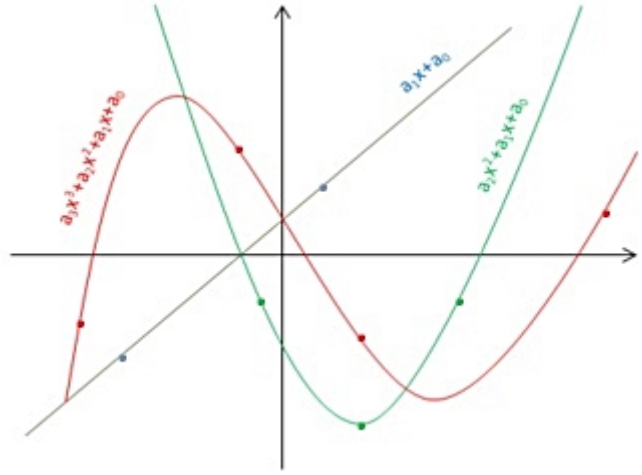
Протокол розділення секретів 2

A_p має секрет $S \in (N)$ Розділення секрету A_p

1. Обираємо велике просте p, $p > s$ (p-відкрита інформація).
2. Обираємо $k-1$ випадкових елементів поля F_p: $a_i \in \{1,...,p-1\}$, $i = \overline{1,k-1}$ та вважають $a_0 = s$ будуємо поліном над F_p. $f(x) = \sum_{i=0}^{k-1} a_i x^i$ степені $k-1$, $a_{k-1} \neq 0$. Числа a_i $i = \overline{1,k-1}$ та поліном - це секрет арбітра A_p
3. Кожному учаснику A_i арбітр A_p передає число $f(i)$, $i = \overline{1,n}$- номер учасника.Таким чином кожен учасник має пару $(i,f(i))$, $i = \overline{1,n}$.

Відновлення секрету S

Нехай зібралися k учасників A_{i_l}, $l = \overline{1,n}$. Приклад: з поліномів $f : R \to R$



$a_1x + a_0$ - відновлюється по 2 точкам;

$a_2x^2 + a_1x + a_0$ - відновлюється по 3 точкам;

$a_3x^3 + a_2x^2 + a_1x + a_0$- відновлюється по 4 точкам.

Розглянемо поліном 2ї степені: складемо систему:

$$\begin{cases} a_2x_1^2 + a_1x_1 + a_0 = y_1, \\ a_2x_2^2 + a_1x_2 + a_0 = y_2, \\ a_2x_3^2 + a_1x_3 + a_0 = y_3. \end{cases}$$

1. Учасники A_{i_l}, $l = \overline{1,k}$ відновлюють поліном $f(x)$ по парам(по k точкам): $\{i_l, f(i_l), l = \overline{1,k}\}$ за допомогою ітераційної формули Лагранжа $f(x) = \sum_{l=1}^{k} f(i_l) \prod_{j \neq l} \frac{x - i_j}{i_l - i_j}$.

2. Обчислюють $f(0) = a_0 = s$.

Чому число учасників $r < k$ про секрет S не дізнаються ніякої інформації?
Нехай зібрались учасники A_{i_l}, $l = \overline{1,k-1}$. Вони мають $k-1$ пару $\{i_l, f(i_l), l = \overline{1,k-1}\}$. (2)
Додамо до множини (2) пару $(0,U_k)$. Тоді через точки $\{(i_l,f(i_l)), i = \overline{1,k-1}, (0,U_k)\}$, $U_k \in F_p$ можна побудувати поліном $\tilde{f}(x)$ степені $k-1$ та у цього поліному $f(0) = U_k$.

4.5 Електронні вибори

(Автор: Михайло Столович. Не редагувалось)
(Версія від 21 січня 2017 р.)

4.5.1 Приклади

Наразі вже декілька розвинутих країн практикують електронні вибори. Тобто, держава дає можливість людині проголосувати за свого кандидата не виходячи з дому.

Можна розглянути, як це реалізовано на прикладі такої країни як Естонія.

Кожному громадянину Естонії, коли йому виповнилося 15 років, видається так звана ID-картка, що виконує функції паспорта в нашому розумінні. Однак, на відміну від звичайного паспорту в неї можна зберігати більше інформації про людину. Її навіть використовують як закордонний паспорт в деяких країнах.

Вже у 2005 році вперше була випробована на реальному голосуванні система електронних виборів. Вона була визнана успішною. Естонія стала першою країною світу, де стало можливим проголосувати через Інтернет.

4.5.2 Вимоги до електронного голосування

Задля того, щоб побудувати систему електронного голосування, спочатку треба визначитися з вимогами, які ця система повинна задовольняти:

1. голосувати може лише зареєстрована людина;
2. кожен може проголосувати лише один раз;
3. усі голоса враховуються анонімно: зберігається таємниця виборів;
4. проголосувавши, ми не маємо змоги змінити своє рішення;
5. неможливість підробки результатів голосування;
6. повинна бути можливість відкрито перевірити результати голосування;
7. ми повинні мати змогу взнати, скільки людей проголосувало.

Як ми бачимо, виставленні вимоги до системи, десь співпадають, а десь перевищують можливості звичайного голосування. Проте давайте розглянемо, як усю цю купу вимог реалізувати на практиці.

4.5.3 Схема електронного голосування

Зазвичай організацією виборів займається якась державна структура. В електронному голосуванні також маємо структуру, яка відповідає за організацію виборів. Назвемо її виборчим центром Σ.

В нашого виборчого центру, як і в кожного чемного виборчого центру існує довірена особа D, яка буде виконувати функції обробки голосів виборців.

Самих виборців позначимо $A_i, i = \overline{1,N}$. Звісно, що в кожного виборця є якась своя унікальна комбінація параметрів D_i, яка його ідентифікує. Також, у кожного виборця є своя RSA пара відкритого ключа (n_i,e_i) і секретного ключа $d_i, i = \overline{1,N}$.

Перед початком самих виборів ми повинні скласти список усіх учасників процесу. Зазвичай, цим займається виборчий центр. Виходячи з тієї інформації, що в нас є логічно зберігати список виборців у вигляді:

$$T = \{< D_i, (n_i,e_i), i = \overline{1,N}\}$$

У довіреного лиця також є своя RSA із відкритим ключем (m,e) і таємним ключем d.

4.5.4 Процедура голосування виборців

Нехай в нас існує r можливих варіантів, за кого голосувати. $r-1$ кандидат і один варіант "Проти всіх".

Розглянемо приклад, коли в нас є наступні кандитати:

1. Фідель
2. Мао
3. Наполеон
4. Нельсон

Спочатку, виборчий центр обирає кодування для кандидатів. Візьмемо варіант, коли $i-$ кандидат представлен $i-$ простим числом. Тобто для нашого варіанту маємо:

- Фідель 2
- Мао 3
- Наполеон 5
- Нельсон 7
- Проти всіх 1

І ось настає славетна година, коли виборець A_i йде на вибори - до компьютера голосувати за свого кандидата b_i.

Спочатку він, як і кожен чемний виборець, сгенерує своє просте число q_i, котре більше за усі числа-коди кандидатів. (У нашому прикладі $q_i > 7$)

Обчислює $t_i = b_i * q_i$ за умови, що $b_i * q_i < m$, де m - це частина відкритого ключа для D.

Після цього він зашифровує свій власний вибір t_i відкритим ключем довіреного лиця D

$$C_i = (t_i)^e mod\ m$$

.

Коли задача конфіденційності вирішена, потрібно ще вирішити задачу цілісності, тому ми підписуємо нашу "бюлетень"цифровим підписом:

$$S_i = (h(D_i||C_i))^{d_i} mod\ m$$

де $i = \overline{1,N}$, а h(.) - геш-функція

І тільки після цього ми передаємо свій зашифрований бюлетень виборчому центру:

$$M_i = (D_i,\ C_i,\ S_i) \rightarrow \Sigma$$

Після того, як усі охочі виборці зроблять свій вибір, вибори закриваються, і ми переходимо до етапу підрахунку голосів.

4.5.5 Підрахунок голосів Σ -ою

Спочатку, довірча особа D перевіряє цифровий підпис кожного виборця $A_i, i = \overline{1,N'}$:

1. Спочатку йде перевірка, чи була особа D_i зареєстрована для участі у виборах.
2. Потім перевіряємо валідність цифрового підпису:

$$h(D_i||C_i)^{e_i}\ mod\ n_i\ ?=?\ h(D_i, C_i)$$

3. Далі, якщо усі перевірки до цього пройдені, то D розшифровує повідомлення:

$$t_i{}^d\ mod\ n = b_i\ *\ q_i$$

   та помічаємо A_i, як проголосувавшого.

4. Останнім кроком D вираховує значення:

$$Q = \prod_{i=1}^{N}\ b_i\ *\ q_i$$

   та відправляє його до виборчого центру Σ.

Виборчий центр розкладує Q на множники

$$Q = 2^{r_1}\ *\ 3^{r_2}\ *\ 5^{r_3}\ *\ 7^{r_4}\ *\ R$$

де $R = \prod_{i=1}^{N'}\ q_i$

І саме з розкладу визначає розподіл голосів по кандидатах.

Після підрахунку результати публікуються у формі:

$$\{\ \{< D_i, C_i >, i = \overline{1,N'}\}\ ,\ r_1,\ r_2, \ldots\ ,\ r_k,\ N - \sum_{i=1}^{N'} r_i,\ R,\ N - N'\}$$

де $N - \sum_{i=1}^{N'} r_i$ - це кількість голосів відданих варіанту "Проти всіх";
$N - N'$ - це кулькість не голосувавших виборців.

4.5.6 Перевірка результатів голосування

1. Кожен виборець має змогу впевнетися, що його голос був зарахований по таблиці результатів. Тобто, воно буде присутнє в опублікованих даних виборів.

   Якщо ж виборець не голосував, то він не знайде себе серед опублікованих зашифрованих повідомлень.

2. Таємниця голосування забезпечується для усіх, окрім довіреного лиця D.

3. Як ми можемо бачити із схеми голосування, проголосувати за когось іншого неможливо. Підробити голос також.

4. Перевірку правильності підрахунку голосів може перевірити будь-яка людина.

   Перш за все, ми перевіряємо те, щоб R не повинно ділитися на 2, 3, 5, 7

   З результатів виборів легко відновлюється число Q. Задля того, щоб перевірити правильність результатів, усе, що нам потрібно - це впевнетися, що має місце такая рівність:

$$Q^e \ mod \ m = (\prod_{i=1}^{N'} C_i) \ mod \ m$$

   Дійсно, якщо усе вірно, то:

$$Q^e \ mod \ m = (\prod_{i=1}^{N'} 2^{r_1} * 3^{r_2} * .. * 7^{r_4} * \prod_{i=1}^{N} q_i)^e \ mod \ m = \prod_{i=1}^{N} (b_i * q_i)^e \ mod \ m = (\prod_{i=1}^{N'} C_i) \ mod \ m$$

4.6 Lecture 16