НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ № 3

з дисципліни:

«ПРОЕКТУВАННЯ, РОЗРОБКА І РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ СИСТЕМ»

Дослідження криптографічних протоколів систем WebMoney, PayPal

Виконала:

Студентка групи ФІ-22мн

Калитюк Дар’я

КИЇВ 2023

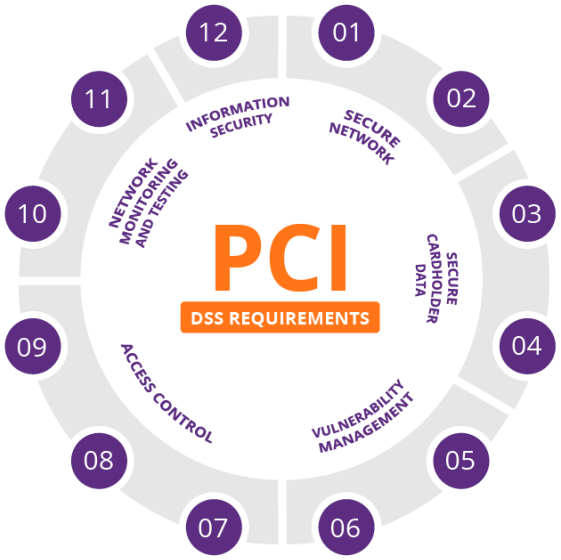
Мета роботи: дослідження особливостей реалізації криптографічних механізмів платіжних систем.

Хід роботи

Електронний гаманець як частина електронної платіжної системи – програмне забезпечення, що дозволяє проводити операції поповнення, зберігання та перерахування електронних грошей, тобто грошей, взаєморозрахунки з яких проводяться за допомогою інформаційних технологій. Очевидно, жодна електронна платіжна система не може існувати без надійних технологій безпеки (методів шифрування, протоколів передачі даних тощо). Порівняємо засоби забезпечення безпеки двох відомих платіжних систем – PayPal та WebMoney.

PayPal — міжнародна електронна платіжна система, яка надає послуги в транскордонному режимі будь-якій людині у світі, у якої є банківська картка.

WebMoney – електронна система миттєвих інтернет-розрахунків. З юридичної точки зору в системі відбувається трансфер майнових прав, облік яких відбувається за допомогою спеціальних розрахункових одиниць – «титульних знаків», які прив'язані до різних валют та золота.



Стандарт безпеки даних індустрії платіжних карток (PCI DSS) — це набір стандартів безпеки, сформований у 2004 році Visa, MasterCard, Discover Financial Services, JCB International і American Express. В PCI SSC окреслено 12 вимог, розподілені між шістьма цілями, щодо обробки даних власників карток і підтримки безпечної мережі:

1. Необхідно встановити та підтримувати конфігурацію мережевого екрану;
2. Системні паролі мають бути оригінальними (не надаватися постачальником);
3. Збережені дані власника картки повинні бути захищені;
4. Передача даних власника картки через загальнодоступні мережі має бути зашифрованою;
5. Необхідно використовувати антивірусне програмне забезпечення та регулярно оновлювати його;
6. Необхідно розробляти та підтримувати безпечні системи та програми;
7. Доступ до даних про власника картки має бути обмежено для ділової необхідності;
8. Кожній особі, яка має доступ до комп'ютера, має бути присвоєно унікальний ідентифікатор;
9. Необхідно обмежити фізичний доступ до даних власника картки;
10. Необхідно відстежувати та контролювати доступ до даних власника картки та мережевих ресурсів;

11. Системи безпеки та процеси необхідно регулярно тестувати;
12. Необхідно підтримувати політику щодо інформаційної безпеки;

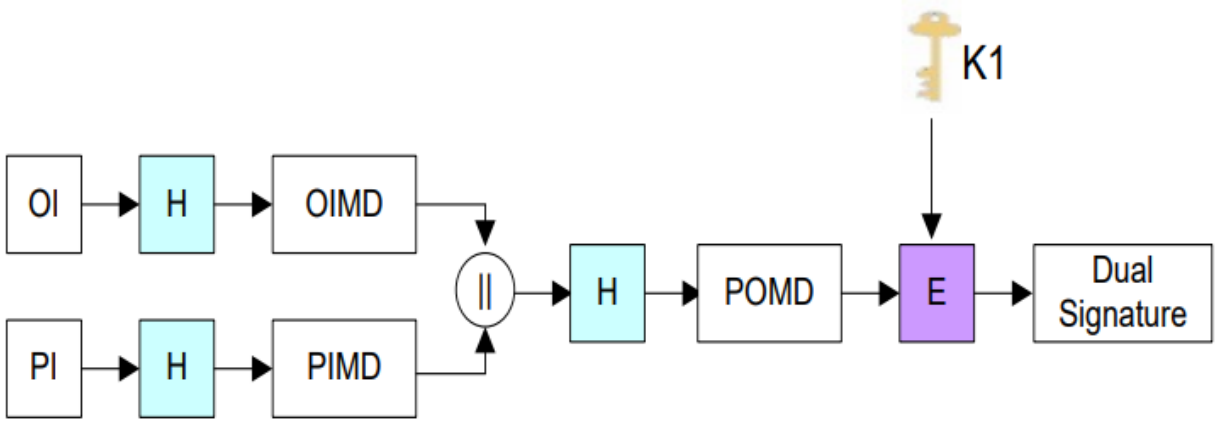
PayPal (246 млн)	WebMoney (45 млн)
RSA 1024 + гешування	RSA 1024 + гешування
2FA автентифікація	2FA автентифікація
HTTPS + TLS	HTTPS + TLS

SET (Secure Electronic Transaction) - це стандарт безпеки електронних платежів, розроблений MasterCard і VISA при значній участі IBM, GlobeSet та інших партнерів.

SET висуває наступні вимоги для захисту електронних платежів:

- Конфіденційність платежів і конфіденційність інформації про замовлення (OI), переданої разом з платіжними даними (PI);
- Підтримка цілісності цих платежів; цілісність забезпечується цифровим підписом;
- Автентифікація кредитної картки та продавця, які забезпечуються електронним підписом і сертифікатами;
- Підтвердження того, що банк продавця є активною організацією, яка може приймати платежі; це підтвердження забезпечується ЕЦП і сертифікатами банку;

В SET також представлена концепція подвійного підпису для поєднання двох фрагментів інформації і для надсилання її двом різним отримувачам: PI для банку, OI для продавця.



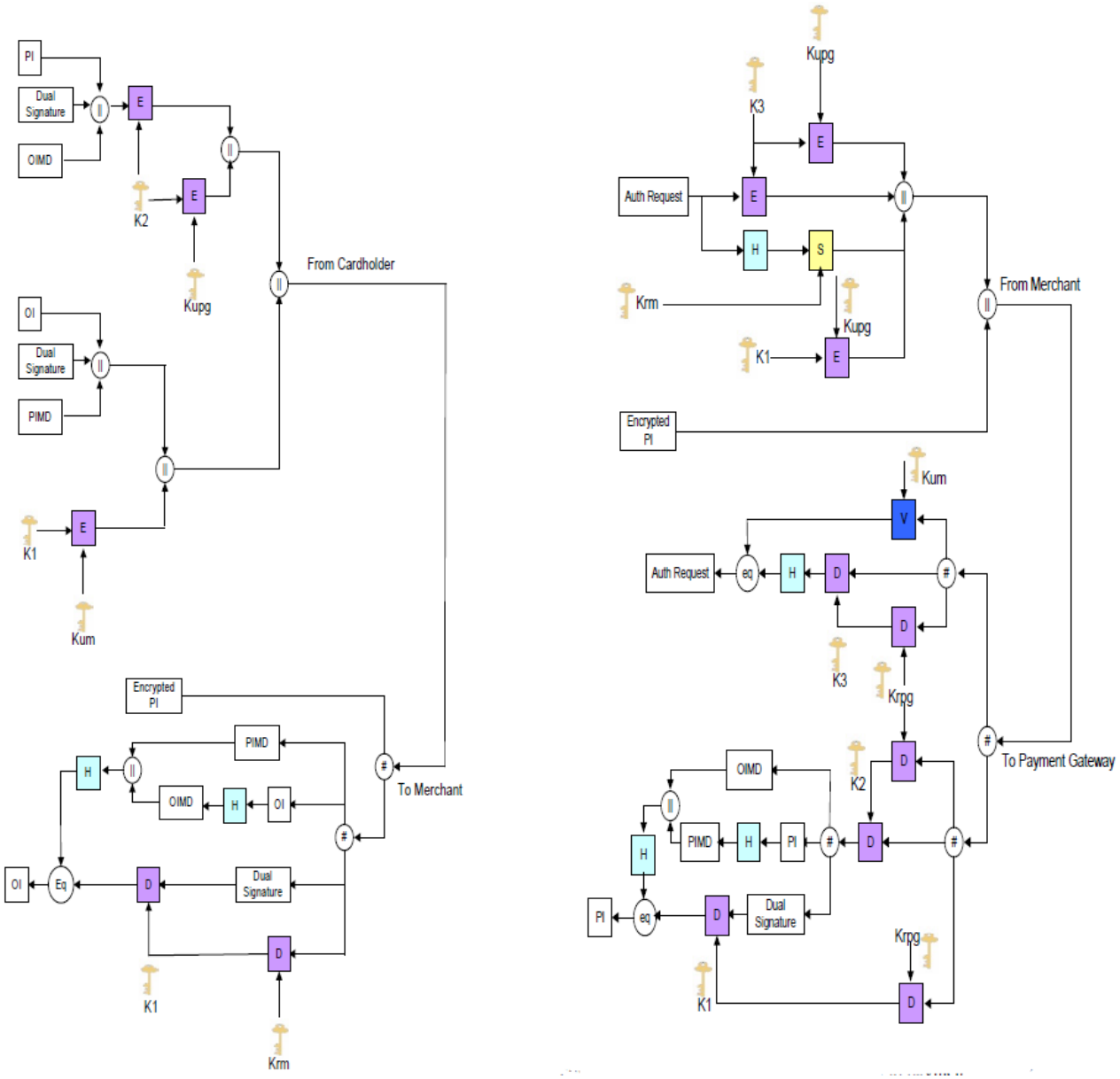
Процес формування дуального підпису можна описати однією формулою:

$$\boldsymbol{DS = E_{K_1}(H(H(PI)||H(OI)))},$$

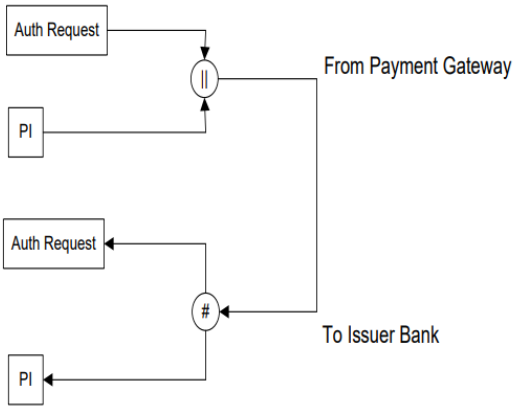
Де K_1 – симетричний ключ клієнта.

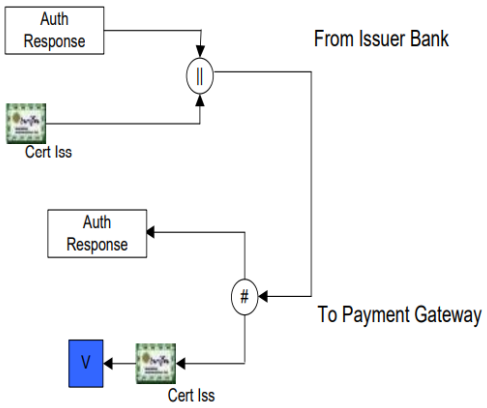
Формування запиту на купівлю від клієнта наведено на малюнку зліва. Тут K_2 – ще один симетричний ключ шифрування, який генерує клієнт, K_{upg}— асиметричний публічний ключ

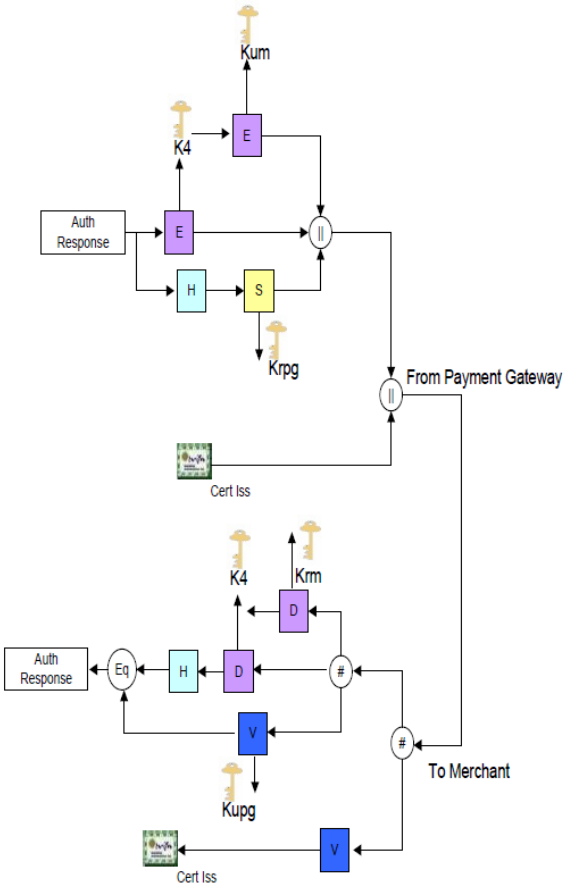
платіжного шлюзу, K_{um}— асиметричний публічний ключ продавця. Продавець отримує весь запит, однак не може дістатись до PI, оскільки не знає асиметричного приватного ключа платіжного шлюзу і не може «розпакувати» конверт із K_2.



Щоб перевірити дуальний підпис і цілісність OI продавець робить наступні крок: розшифровує симетричний ключ K_1 за допомогою свого приватного асиметричного ключа K_m, розшифровує підпис і обчислює POMD. Продавець генерує, шифрує та підписує запит авторизації та отримані від клієнта зашифровані дані для отримання PI та конверт із значенням K_2. Платіжний шлюз перевіряє підпис продавця, розпаковує ключ K_2, розшифровує PI nf по аналогічній схемі перевіряє дуальний підпис.



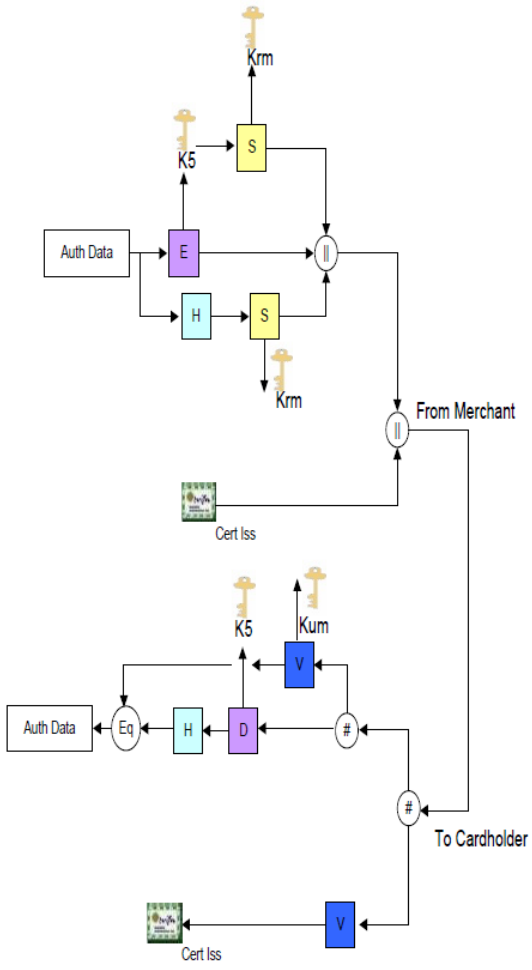




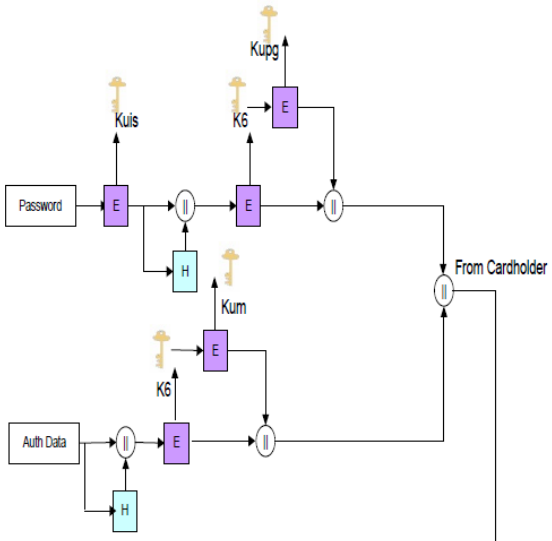
симетричний ключ і перевіряє цілісність відповіді автентифікації.

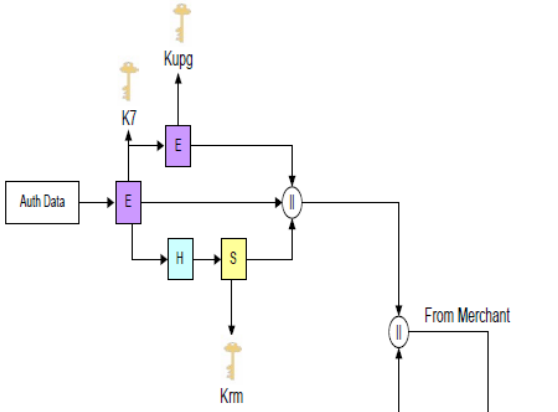
Якщо *action code* = "Y", продавець надсилає запит автентифікації до клієнта, який містить також сертифікат банку. Клієнт перевіряє дані і сертифікат та надсилає продавцю свій зашифрований асиметричним ключем банку а потім симетричним пароль. Продавець далі надсилає його на платіжний шлюз, де платіжний шлюз розшифровує розпакованим

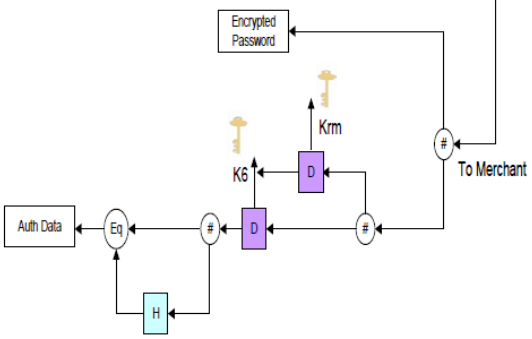
Далі платіжний шлюз передає PI банку. Банк перевіряє дані запиту автентифікації, PI і надсилає відповідь верифікації, що містить значення *response code* (чи погоджено запит авторизації) і *action code* (чи треба клієнту пройти автентифікацію паролем) та свій сертифікат платіжному шлюзу для верифікації. Платіжний шлюз шифрує та підписує відповідь верифікації і надсилає продавцю разом із сертифікатом. Продавець перевіряє сертифікат, за вже відомою схемою розпаковує

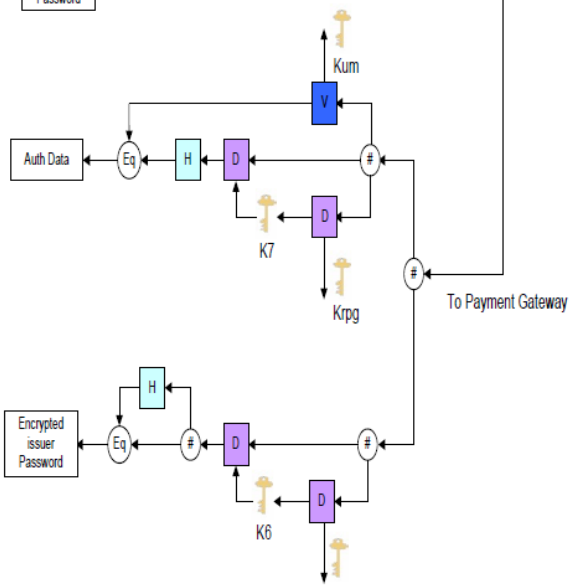


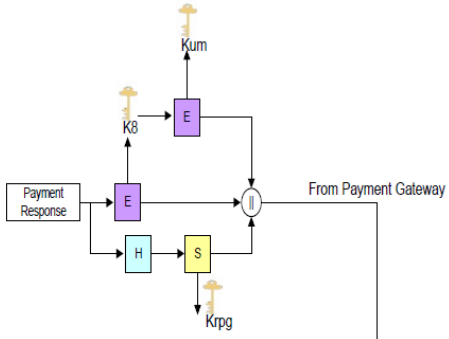
ключем пароль, перевіряючи його цілісність, і відправляє в банк (пароль все ще зашифрований ключем банку).



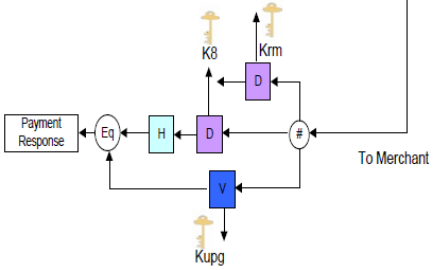


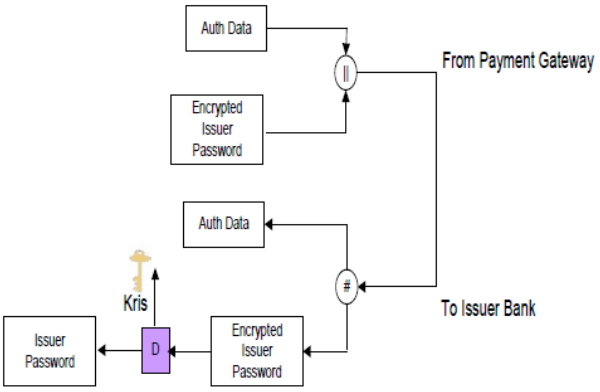






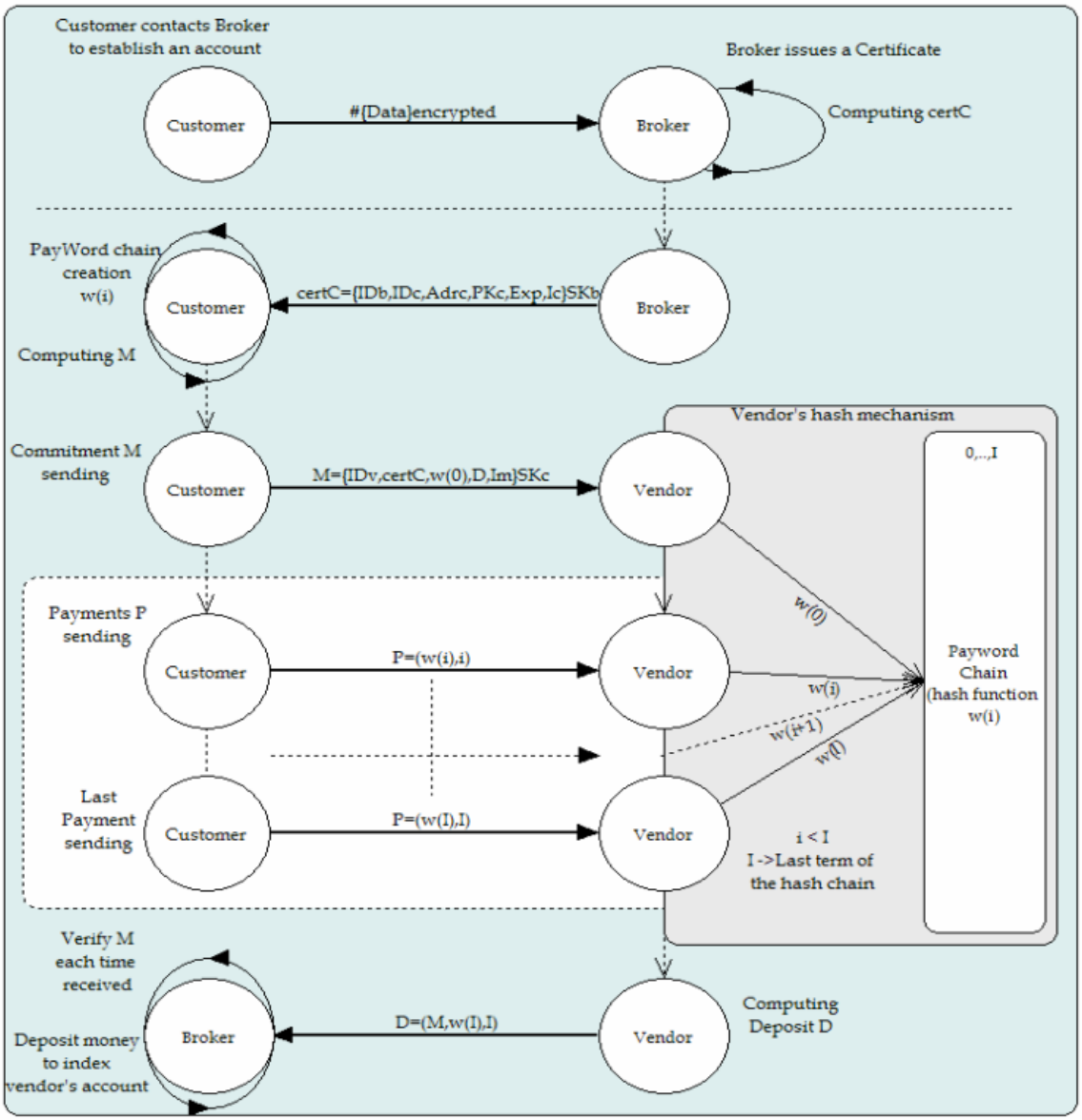
Банк розшифровує та перевіряє пароль і надсилає відповідь платіж платіжному шлюзу, який своєю чергою, передає його продавцю. Після перевірки відповіді продавцем угода завершується і продавець може надсилати клієнту товар.





PayWord і **MicroMint** – це дві системи мікроплатежів, створені Рональдом Рівестом та Аді Шаміром. Їхньою головною метою при створенні даних схем була мінімізація кількості використань операції з публічним ключем і заміна їх на гешування задля досягнення високої ефективності роботи алгоритмів.

PayWord — це офлайн протокол мікроплатежів, реалізований за допомогою геш-ланцюжків, які називаються ланцюжками *PayWords.* В протоколі задіяно три учасники: клієнт, брокер і постачальник. Клієнт створює обліковий запис у брокера, який видає йому сертифікат. Цей сертифікат дозволяє клієнту створювати ланцюжки *Paywords* і автентифікує клієнта для постачальника.



Робота PayWord починається з того, що клієнт запитує у брокера акаунт і сертифікат, надсилаючи свої зашифровані дані: номер кредитної карти, публічний ключ PK_C і свою адресу Adr_C. Сертифікат клієнта містить наступні поля: імена брокера і клієнта, PK_C, Adr_C, термін придатності сертифікату Exp та іншу інформацію про клієнта I_C.

При першому запиті клієнта до нового постачальника він обчислює ланцюжок *PayWords* $w_1, w_2, \dots, w_n$, де w_n обирається випадкового, наступним чином:

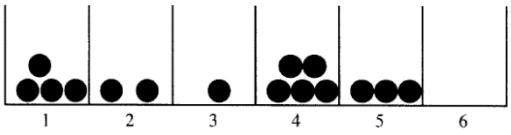
$$w_i = h(w_{i+1}), \qquad i = \overline{n-1, 0}$$

Зауважимо, що w_0 називається коренем ланцюжка і не є *PayWord* за визначенням. Після цього клієнт надсилає постачальнику підписане своїм секретним ключом SK_C зобов'язання M, в якому містяться ім'я постачальника, сертифікат клієнта, w_0, поточна дата D і додаткова інформація I_m. Постачальник перевіряє підпис зобов'язання і сертифікат клієнта.

Кожен платіж являє собою пару $P = (w_i, i)$. Якщо припустити, що платежі однакові і дорівнюють, наприклад, 1 центу, то постачальнику необхідно зберігати у себе тільки значення w_0 та w_n, де n – кількість центів, які постачальник має отримати від клієнта.

Після того, як клієнт надіслав всі платежі, постачальник надсилає брокеру повідомлення про викуп D, яке містить в собі зобов'язання M і останній платіж – (w_n, n).

MicroMint же не використовує жодної операції з відкритим ключем. Кожна монетка тут представлена k-кратною колізією $\{x_1, \dots, x_k\}$, тобто $h(x_1) = \cdots = h(x_k) = y$. В своїй роботи Рівест і Шамір пропонували розглядати процес обчислення геш-функції як процес випадкового кидання кульки x в одну з 2^n ячеєк. Тоді y – номер ячейки, в якій опинилась кулька. Таким чином, щоб в одній ячейці опинилось k кульок, необхідно розкидати приблизно $k2^n$ кульок. З кожної ячейки, в якій опинилось $\geq k$ кубиків, брокер зможе згенерувати одну монетку. Більше того, для зменшення затрат пам'яті для генерування монет, брокер обирає лише ті y (тобто ячейки), перші t біт яких дорівнюють деякому z, яке змінюється кожен місяць. Вибір t має балансувати обчислювальні витрати та витрати пам'яті: завелике значення t сильно сповільнить швидкість облислення, в той час як замале значення вимагатеме від брокера дуже багато пам'яті.



Отже, за сценарієм Рівеста і Шаміра, брокер кожен місяць генерує доцільну кількість монет (залежно від своїх можливостей і побажань), а на початку нового місяця продає їх клієнтам. Коли клієнт надсилає монети постачальнику, він може легко її валідувати, обчисливши і порівнявши всі геші. В решті решт постачальник знов повертає монети брокеру: той перевіряє, чи не повертались вони раніше, і сплачує постачальнику за монети.