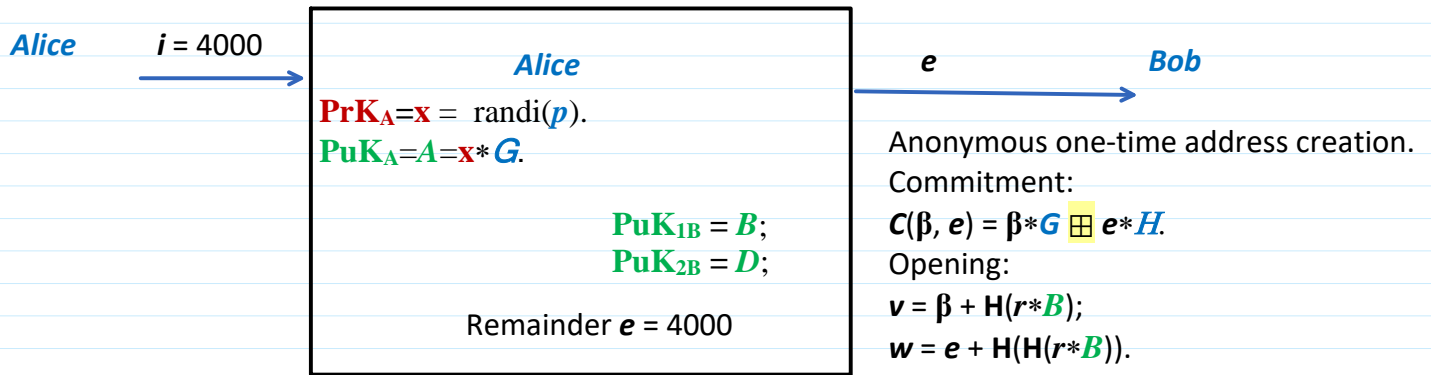


$PP = (EC \text{ secp256k1}; \text{BasePoint-Generator } G; \text{prime } p; \text{param. } a, b);$
 Parameters a, b defines EC equation $y^2 = x^3 + ax + b \pmod p$ over F_p .
 $PrK_A = x;$
 $\gg x = \text{randi}(p).$
 $PuK_A = A = x * G.$
 Alice $A: x = \dots; A = (x_A, y_A);$

Commitments and their opening

Public Parameters: $PP = (G, H)$, where $H = u * G$ and $u \leftarrow \text{randi}(2^{256})$ is random.



Anonymous One-time addresses

Alice: Has **Bob's** public keys $PuK_{1B} = B;$ $PuK_{2B} = D;$
Bob: $y \leftarrow \text{randi}(p); PrK_{B1} = y; PuK_1 = B = y * G = (x_B, y_B);$
 $z \leftarrow \text{randi}(p); PrK_{B2} = z; PuK_2 = D = z * G = (x_D, y_D);$

The first **Bob's** private key y is often called the **view key**.

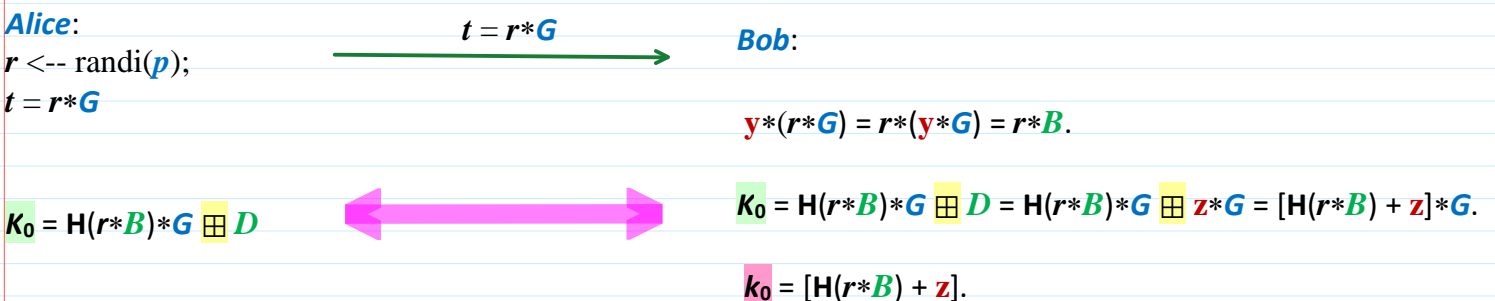
To achieve transaction anonymity by spending an expenses the one-time **address of the payment** is created between the sender **Alice** and receiver **Bob**.

This address is secret (to provide anonymity) and is named also as **one-time key** and is similar to the secret session key agreed by parties.

Let u, v are integers $< p$.

Property 1: $(u + v) * P = u * P + v * P$ in literature it is replaced to $\rightarrow (u + v)P = uP + vP$

Property 2: $(u) * (P + Q) = u * P + u * Q$ in literature it is replaced to $\rightarrow u(P + Q) = uP + uQ$



According to the **PrK** and **PuK** definition in **ECC** k_0 is a private key for the public key K_0 .
 K_0 is the shared secret and is named as *address of the payment* and it is anonymous for the **Net**.
 k_0 is also *one-time key* created for every transaction and corresponding to the private key k_0 .
Neither k_0 nor K_0 are not known the **Net** yet.

Commitments and their opening.

1. Using H-functions: bitcoin price p and salt s .
2. Pedersen commitment: blinding factor β amount of expenses e , hiding, opening.

1. Commitment and its opening using H-functions.

Alice: predicts the bitcoin price p next month and tells it to **Bob**.

Bob: asks **Alice** to say this price.

Alice: said that she is no intending to reveal this knowledge for free.

Bob: promised a reward.

Alice: randomly generates salt $s \leftarrow \text{randi}(2^{256})$

computes $h = H(p \parallel s)$

sends h to **Bob**.

After 1 months bitcoin prices grew up by 510 %

Bob: sold the bitcoins with a great profit and asks to prove its knowledge.

Alice: sends salt s and p to **Bob**.

Bob: verifies if $h = H(p \parallel s)$ and sends **Alice** reward.

2. Pedersen commitment and its opening using ECC.

All users have two generators in **EC**: G and H .

Alice: computes the commitment $C(e)$ to expense value $e = 4000$;

Alice: randomly generates secret blinding value $\beta = \leftarrow \text{randi}(2^{256})$

$$C(\beta, e) = \beta * G \boxplus e * H.$$

Alice: sends $C(\beta, e)$ to **Bob**, to **Net** and to Audit Authority - **AA** which is Trusted Third Party - **TTP**.

Alice: computes *mask* and *expenses* parameters (v, w) respectively and sends to **Bob** by *secret channel* to open the commitment

$$v = \beta + H(r * B);$$

$$w = e + H(H(r * B)).$$

Bob: has previously computed $r * B$ using $y * (r * G) = r * (y * G) = r * B$;

he computes $H(r * B)$ and $H(H(r * B))$;

then he computes:

$$\beta = v - H(r * B);$$

$$e = w - H(H(r * B)).$$

Bob: having public parameters verifies if previously received commitment $C(\beta, e) = \beta * G \boxplus e * H$ is valid.

Bob: Using *one-time key* k_0 agreement signs the expense e and sends signature to **AA**.

$\mathcal{B} : PrK_{AA}, G(\beta, e), \beta, e, K_0.$

$$\text{Sign}(k_0, G(\beta, e)) = \tilde{\sigma}_B = (r_B, s_B)$$

$k \leftarrow \text{randi}$

$$\text{Enc}(PrK_{AA}, k) = c_k$$

$$\text{AES}_k(K_0, G(\beta, e), \beta, e, K_0) = G_B \xrightarrow[\tilde{\sigma}_B]{G_B}$$

$AA : PrK_{AA}, PuK_{AA}$

$$\text{Dec}(PrK_{AA}, G_B) = (K_0, G(\beta, e), \beta, e, K_0)$$

$$K_0 = k_0 * G$$

$$\text{Ver}(K_0, \tilde{\sigma}_B, G(\beta, e)) = \mathbf{T}$$

By having β, e computes e .

Till this place

We can then define the commitment of an amount a as $C(x, a) = xG + aH$, where x is a blinding

Terminology summary

- A **hiding** commitment does not allow an adversary to know what value was selected by the committer. This is usually accomplished by including a random term that the attacker cannot guess.
- A **blinding** term is the random number that makes the commitment impossible to guess.
- An **opening** is the values that will compute to the commitment.
- A **binding** commitment does not allow the committer to compute a hash with different values. That is, they cannot find two (value, salt) pairs that hash to the same value.

From <https://www.rareskills.io/post/pedersen-commitment>

Why the committer must not know the discrete logarithm relationship between B and G

Suppose the committer knows b such that $B=bG$.

In that case, they can open the commitment

$$\text{commitment} = vG + sB$$

to a different (v', s') other than the value they originally committed.

Here's how the committer could cheat if they know that b is the discrete logarithm of B . $B=bG$

The committer can rewrite the commitment equation: $\text{commitment} = vG + sB = vG + s(bG)$ (substituting $B = bG$) $= (v+sb)G$

The committer picks a new value v' and computes s' :

$$v' + s'b = v + sb \implies v' = v + sb - s'b$$

Then, the prover presents (v', s') as the forged opening.

This works because $\text{commitment} = v'G + v + sb - v'b$

$$G\text{commitment} = v'G + vG + s(bG) - v'$$

$G\text{commitment} = vG + sB$ $\text{commitment} = \text{commitment}$ **Therefore, the committer must not know the discrete logarithm relationship between the elliptic curve points they are using.**

One way to accomplish this is to have a verifier supply the elliptic curve points for the committer. A simpler way, however, is to pick the elliptic curve points in a random and transparent way, such as by pseudorandomly selecting elliptic curve

points. Given a random elliptic curve point, we do not know its discrete logarithm.

For example, we could start with the generator point, hash the x and y values, then use that to seed a pseudorandom but deterministic search for the next point.

From <<https://www.rareskills.io/post/pedersen-commitment>>