- comb method of scalar multiplication (exponentiation in $\mathbb{F}_q^*$)
- trades storage for speed
  for fixed g. (or fixed basepoint P)
- implemented e.g. in embedded systems
- mbedtls
  ↖ optimization for EC

We will use multiplicative notation:

$$y = g^x \quad \Big\} \rightarrow Y = x \cdot P$$

↖ arithmetic specific to the multiplicative group of field $\mathbb{F}_q^*$

The algorithm assumes that g is known in advance and some powers of g can be precomputed and stored in the memory.

In order to compute the exponentiation g^e the l-bit exponent is divided into h blocks e_i, the length of a single block is

$$a = \left\lceil \frac{l}{h} \right\rceil$$

The exponent e can be written as,

$$e = e_{h-1} e_{h-2} \cdots e_1 e_0 = \sum_{i=0}^{h-1} e_i (2^a)^i$$

Each block e_i, $i = 0, 1, \ldots, h-1$ is further

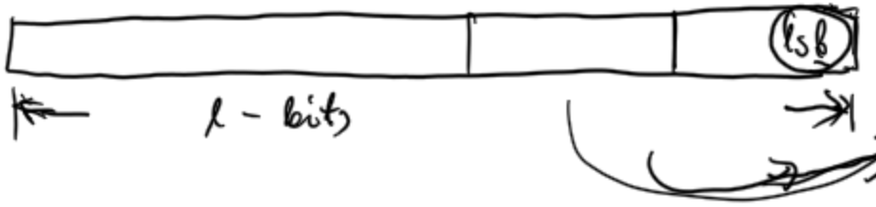
subdivided into v smaller blocks of length $b = \lceil \frac{a}{v} \rceil$ bits.

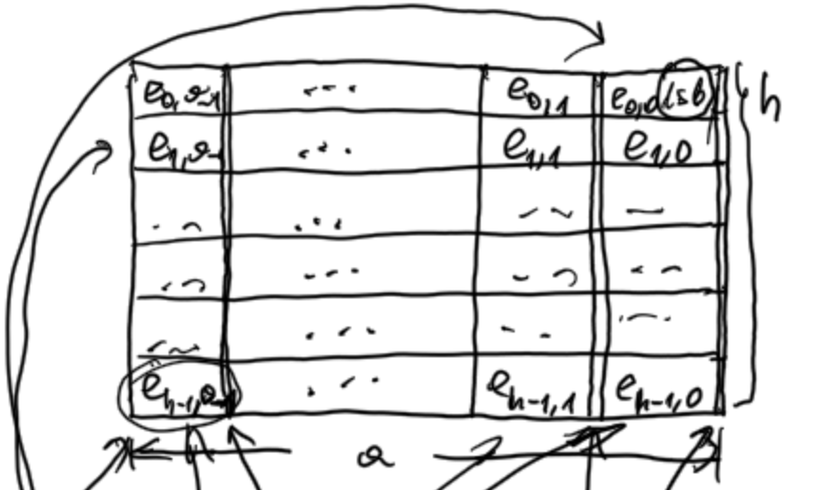
Hence each block e_i can be represented as:

$$e_i = e_{i,v-1}\, e_{i,v-2} \cdots e_{i,1}\, e_{i,0} = \sum_{j=0}^{v-1} e_{i,j} (2^b)^j$$

and each block $e_{i,j}$ composed of b-bits can be expressed as

$$e_{i,j} = e_{i,j,b-1}\, e_{i,j,b-2} \cdots e_{i,j,1}\, e_{i,j,0} = \sum_{k=0}^{b-1} e_{i,j,k} 2^k$$





in each layer we have v subblock

$e_{h-1,v-1}$

all possible values for these bit columns will be pre-computed

In each <u>vertical block</u> $e_{\bullet,0}, e_{\bullet,1}, \ldots, e_{\bullet,2^h-1}$ all possible non-zero value (2^h-1 in total) corresponding to bit $k=0$ will be pre-

$i = 0, 1, \ldots, b-1$

computed. Note that the precomputation can be done in advance, to be prepared to each possible value of e.

When exponent e is known the precomputed values shall be used and a variant of the square-and-multiply method is used.

That is we pre-compute an array

$$G[j][u] \quad \text{with } 0 \le j \le v \quad 1 \le u < 2^b$$

with the entries defined as follows:

- let the binary representation of u be $u_{h-1} u_{h-2} \dots u_1 u_0$
- define r_i as $r_i = g^{2^{ia}} = g^{(2^a)^i}$ for $i = 0, \dots, h-1$

↑ the value of exponentiation corresponding to the lsb of layer i

- then

$$G[0][u] := r_{h-1}^{u_{h-1}} \cdot r_{h-2}^{u_{h-2}} \cdot \dots \cdot r_1^{u_1} r_0^{u_0}$$

$$G[j][u] := \left(G[j-1][u]\right)^{2^b} = \left(G[0][u]\right)^{2^{j \cdot b}}$$

$$j = 1, 2, \dots v-1$$

Define:

$$I_{j,k} = \sum_{i=0}^{h-1} e_{i,j,k} 2^i$$

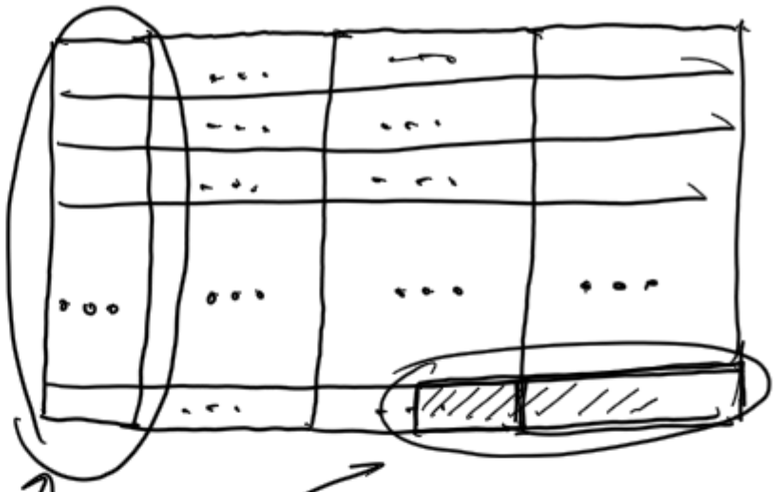
Then the exponentiation can be described as follows:

neutral group element

1. SET $R := 1$
2. FOR $k = b-1$ DOWNTO 0
   (a) SET $R := R^2$
   (b) FOR $j := a-1$ DOWNTO 0
       SET $R := R \cdot G[j][I_{j,k}]$
3. RETURN R

The algorithm above needs $b-1$ squarings and $\frac{2^h - 1}{2^h} a - 1$ multiplications on average and $a-1$ multiplications in the worst case.

However, the vertical blocks $e_{0,a-1}$ can be shorter, as well as the layer e_{h-1}



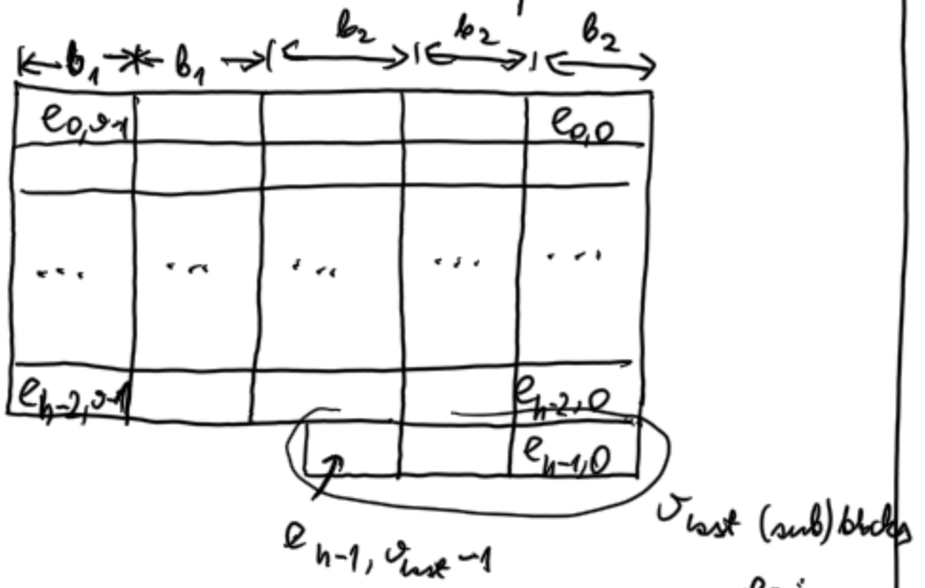
The question is "Can we choose the parameter a, b with greater flexibility?"

The answer is "YES"

In the original paper by Lim/Lee there is a second version of the algorithm, where two blocks widths b_1, b_2 are allowed.

Let h, a be the input parameters as before.

Let the last block e_{h-1} is composed of v_{last} (sub)block $e_{i,j}$



To establish v_{last} we may take it from the first estimation on the basis of the first version of the algorithm:

$$e_{last} = l - a \cdot (h-1)$$ - the bit length of the layer e_{h-1}

$$v_{last} := \left\lceil \frac{e_{last}}{b} \right\rceil$$

Having h, v, v_{last} we may calculate b_1 and have two different widths b_1, b_2:

$$b_2 := \left\lceil \frac{l}{(h-1)v + v_{last}} \right\rceil$$

$$b_1 := \left\lceil \frac{l - b_2 \cdot h \cdot v_{last}}{(h-1) \cdot (v - v_{last})} \right\rceil$$

the number of bits consumed by v_{last} vertical blocks $e_{0,0}, \dots, e_{0,v_{last}-1}$