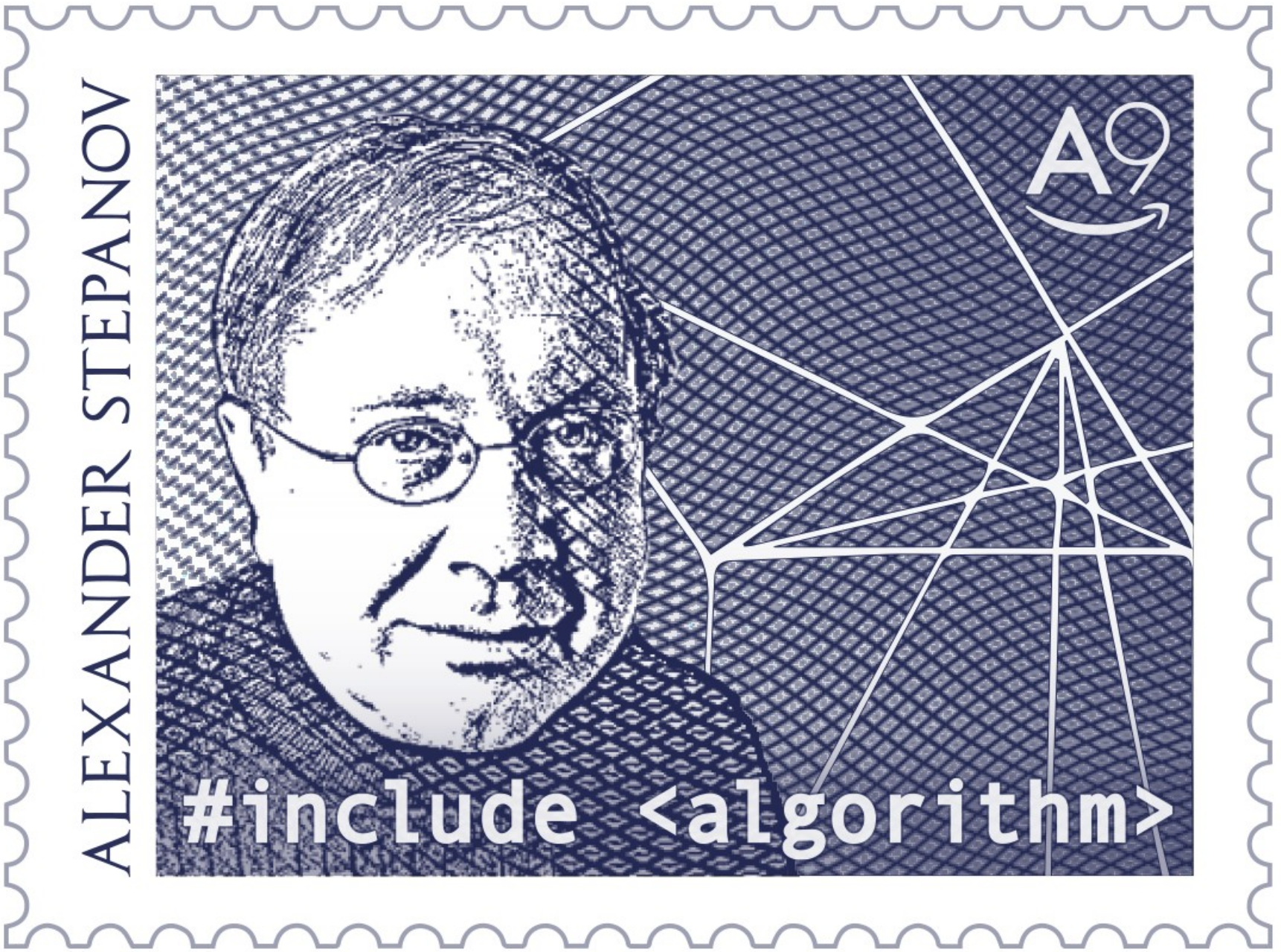
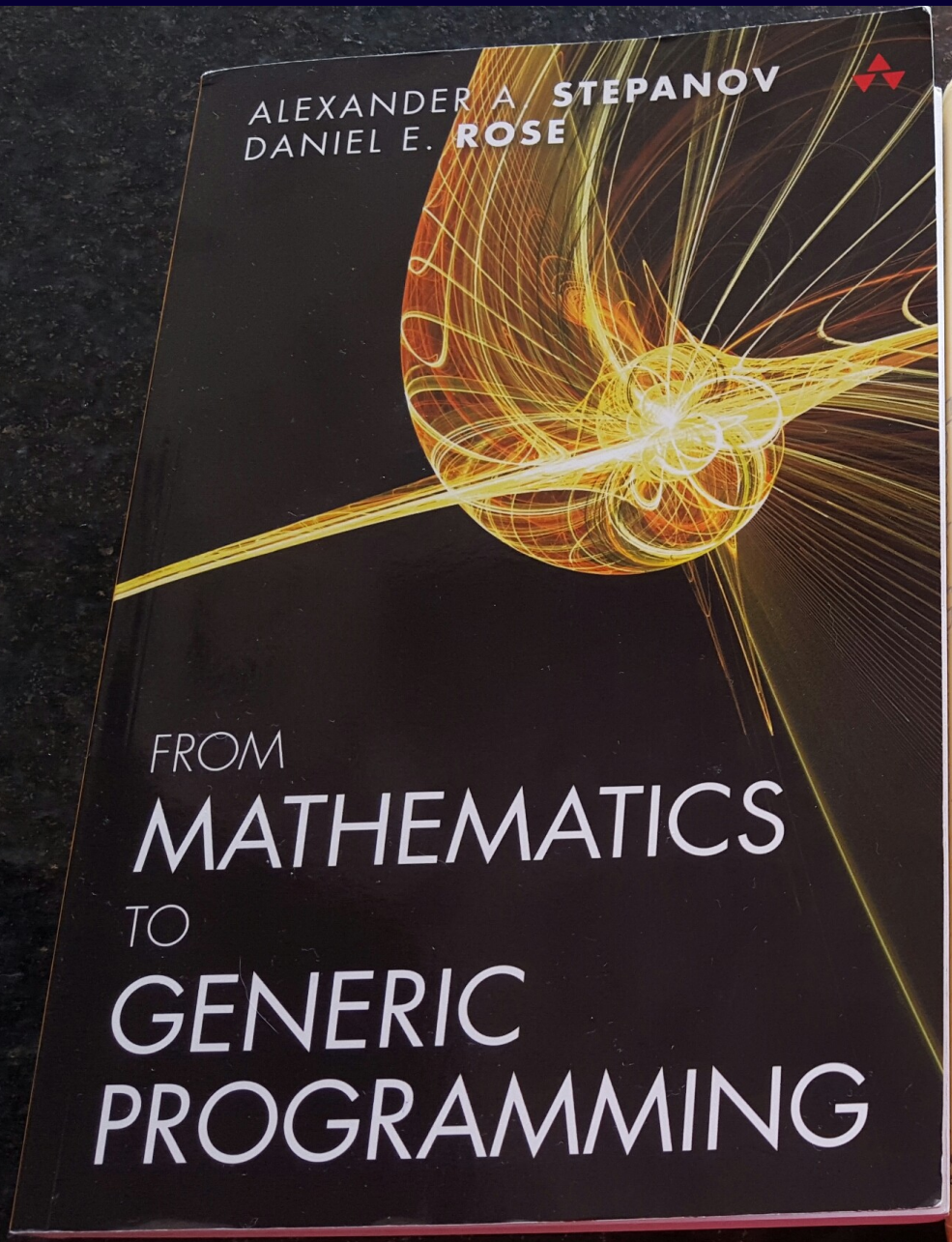
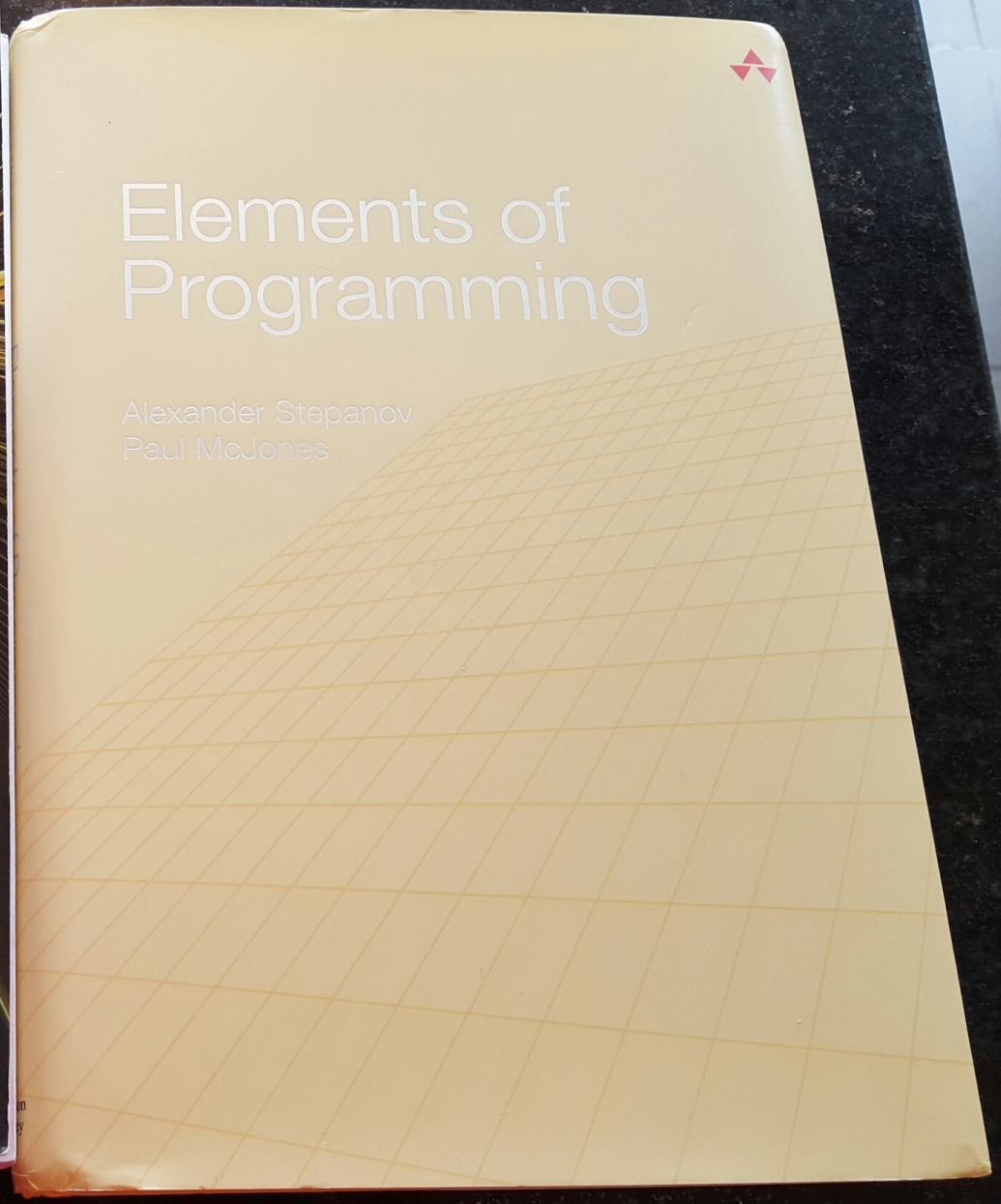

El regalo de los Árabes
@ferpelliccioni


ALEXANDER STEPANOV
#include <algorithm>

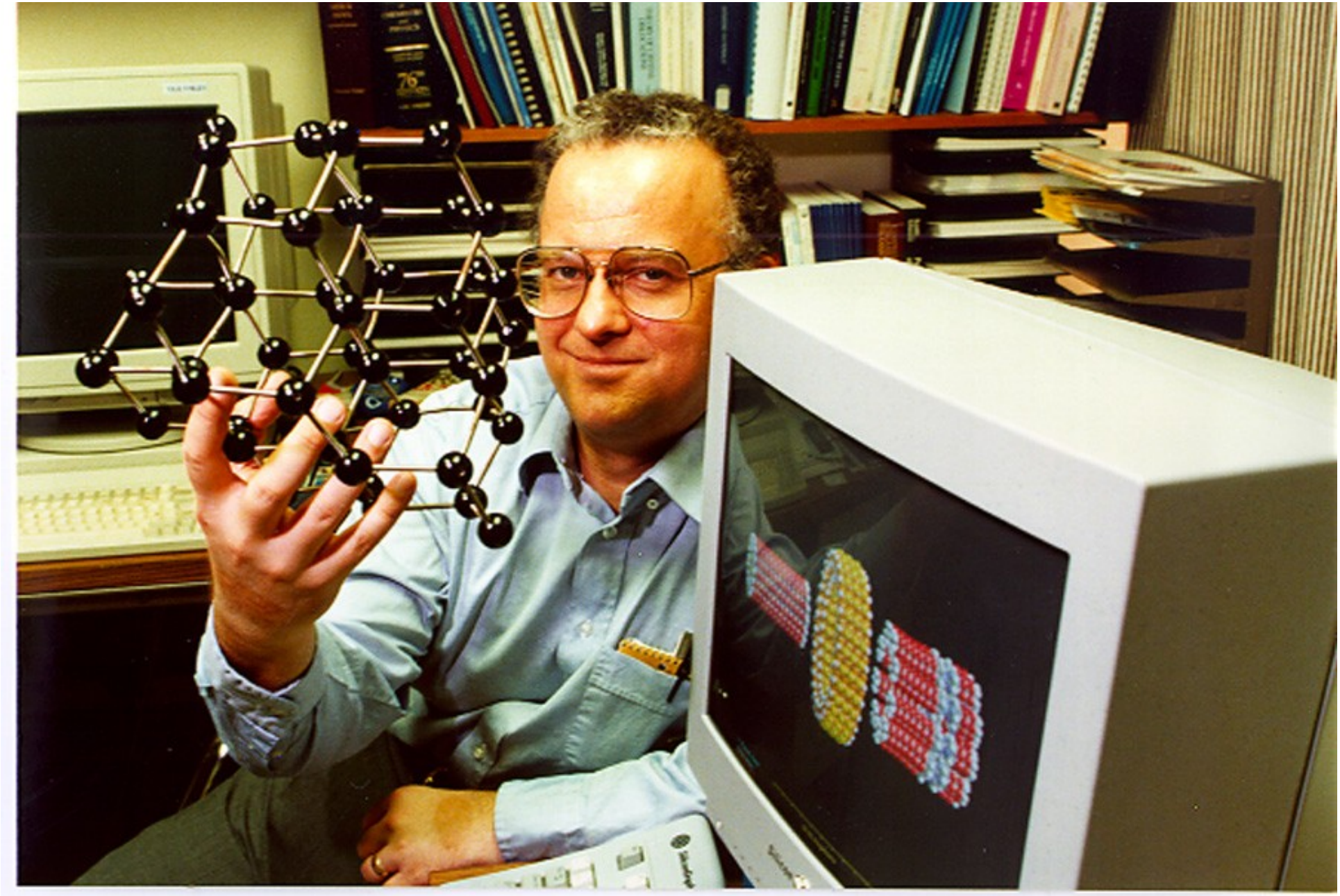
Four Algorithmic Journeys:

I. Spoils of the Egyptians
II. Heirs of Pythagoras
III. Successors of Peano
IV. ????????

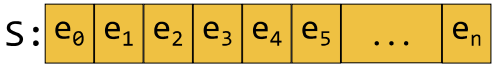

ALEXANDER A. STEPANOV
DANIEL E. ROSE
FROM
MATHEMATICS
TO
GENERIC
PROGRAMMING

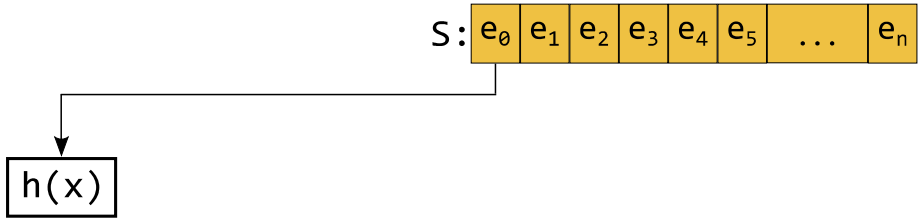

Elements of
Programming
Alexander Stepanov
Paul McJones

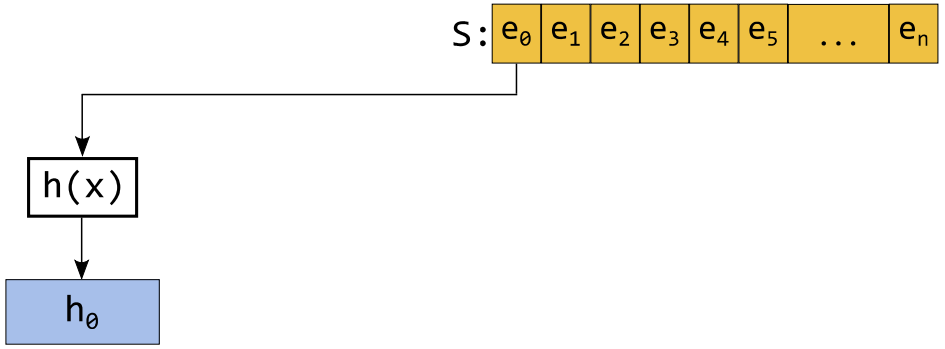
Merkle Tree

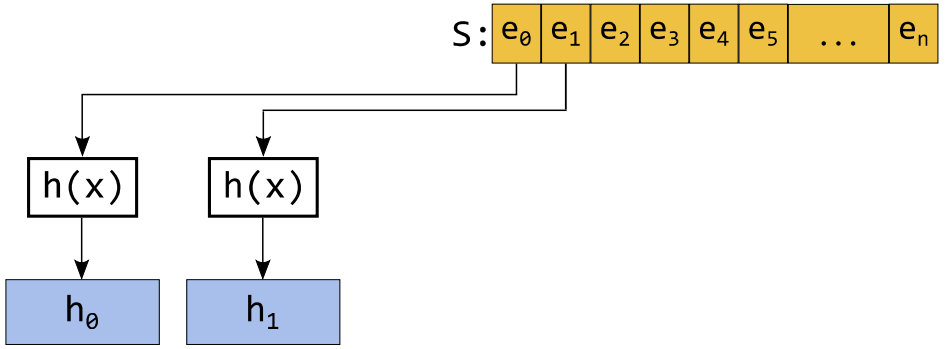


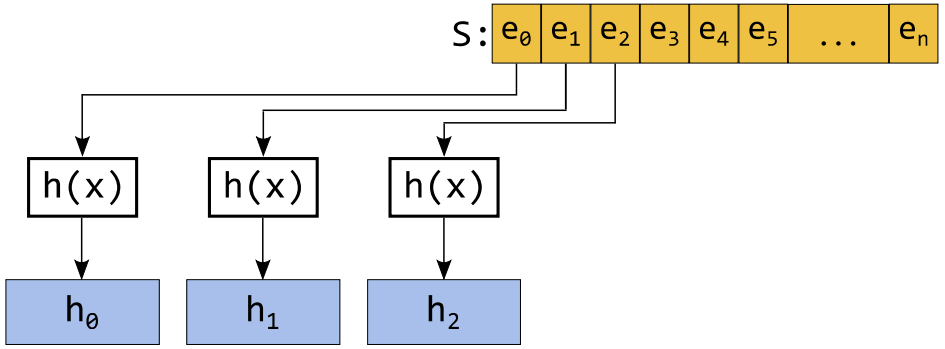
Dr. Ralph C. Merkle

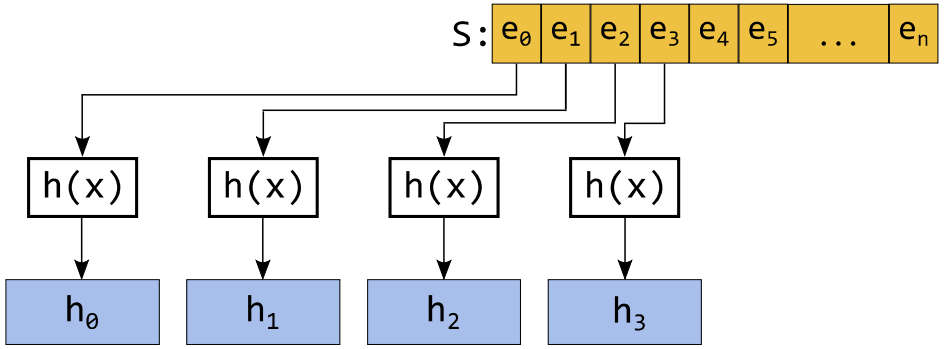

s:
e0
e1
e2
e3
e4
e5
...
en

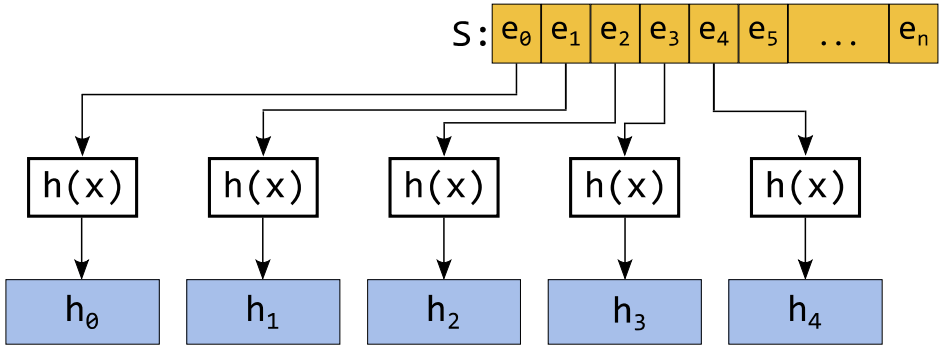

S:
e0
e1
e2
e3
e4
e5
...
en
h(x)

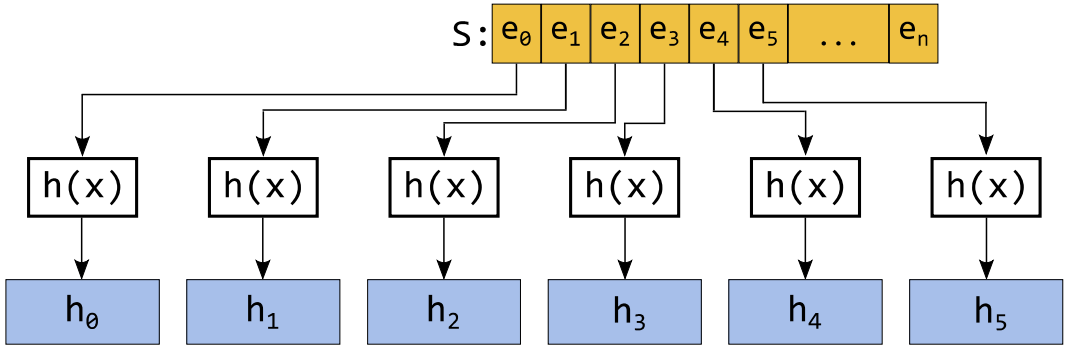

S:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h0

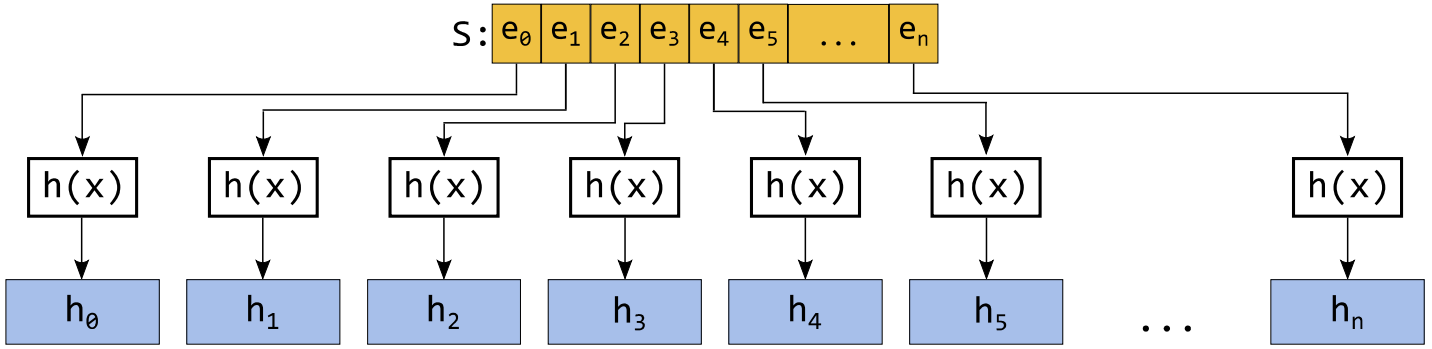

S:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h0
h1

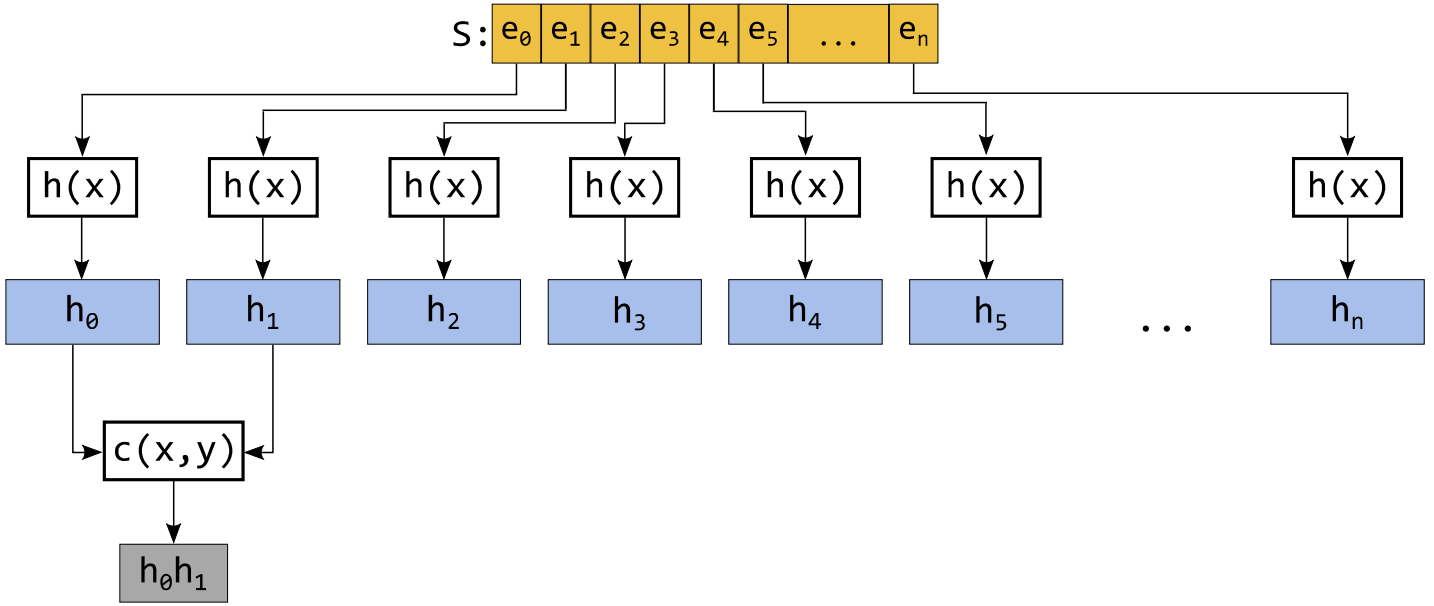

s:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h(x)
h0
h1
h2

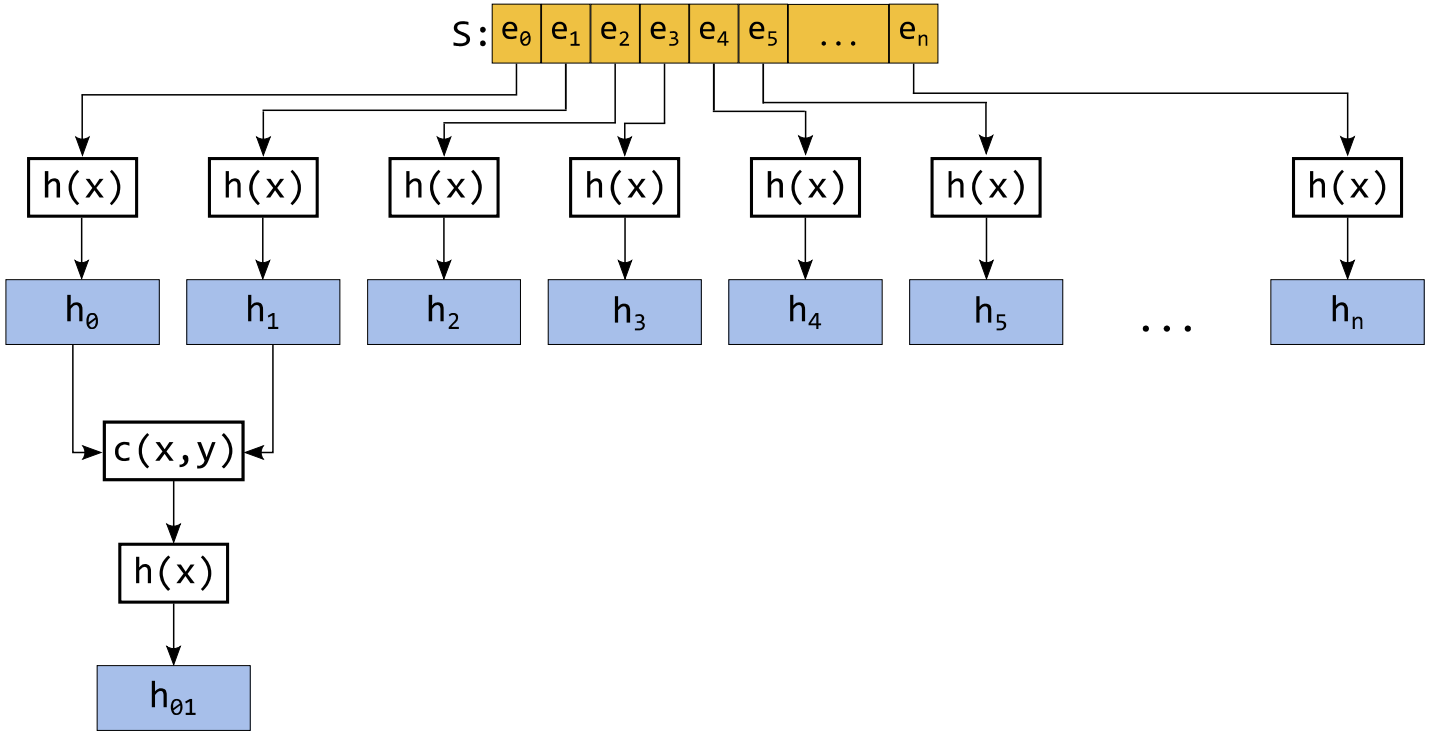

S:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3

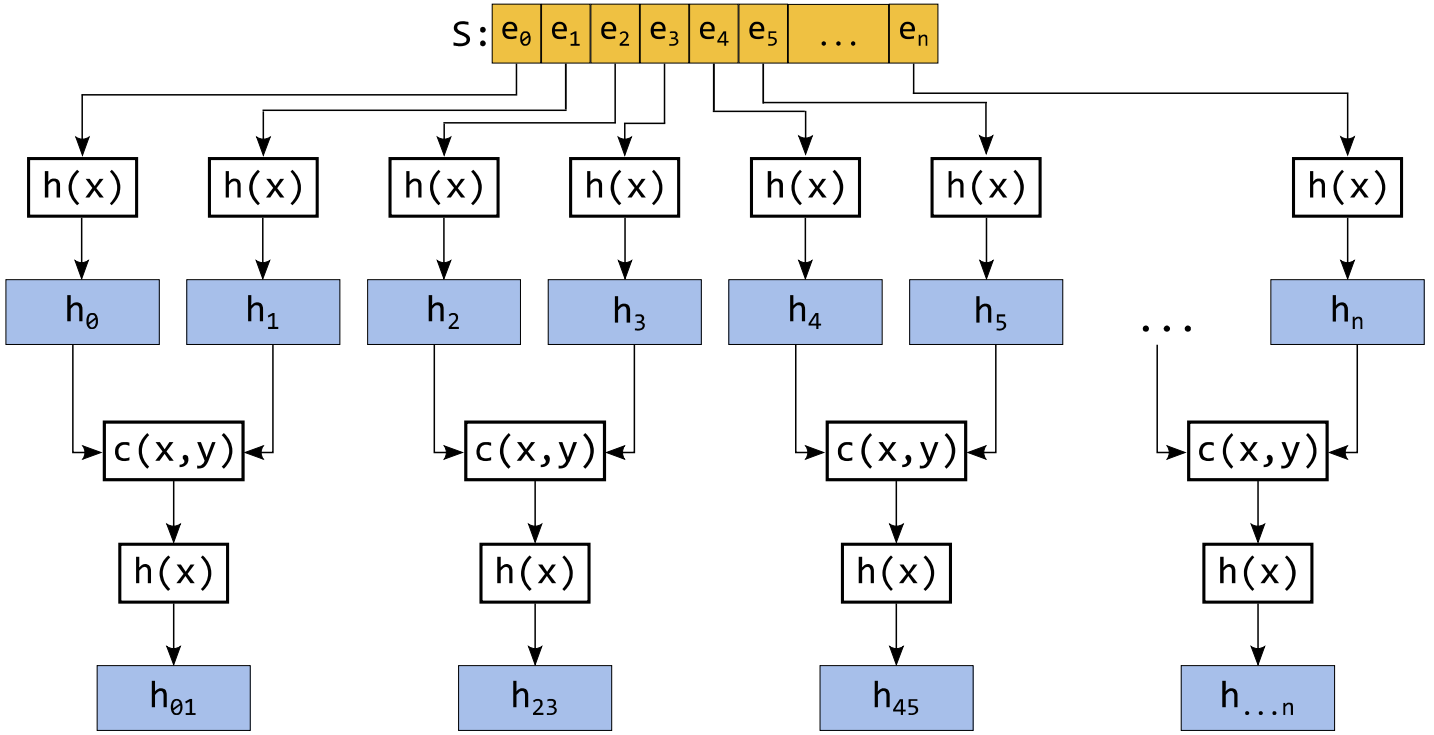

s:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4

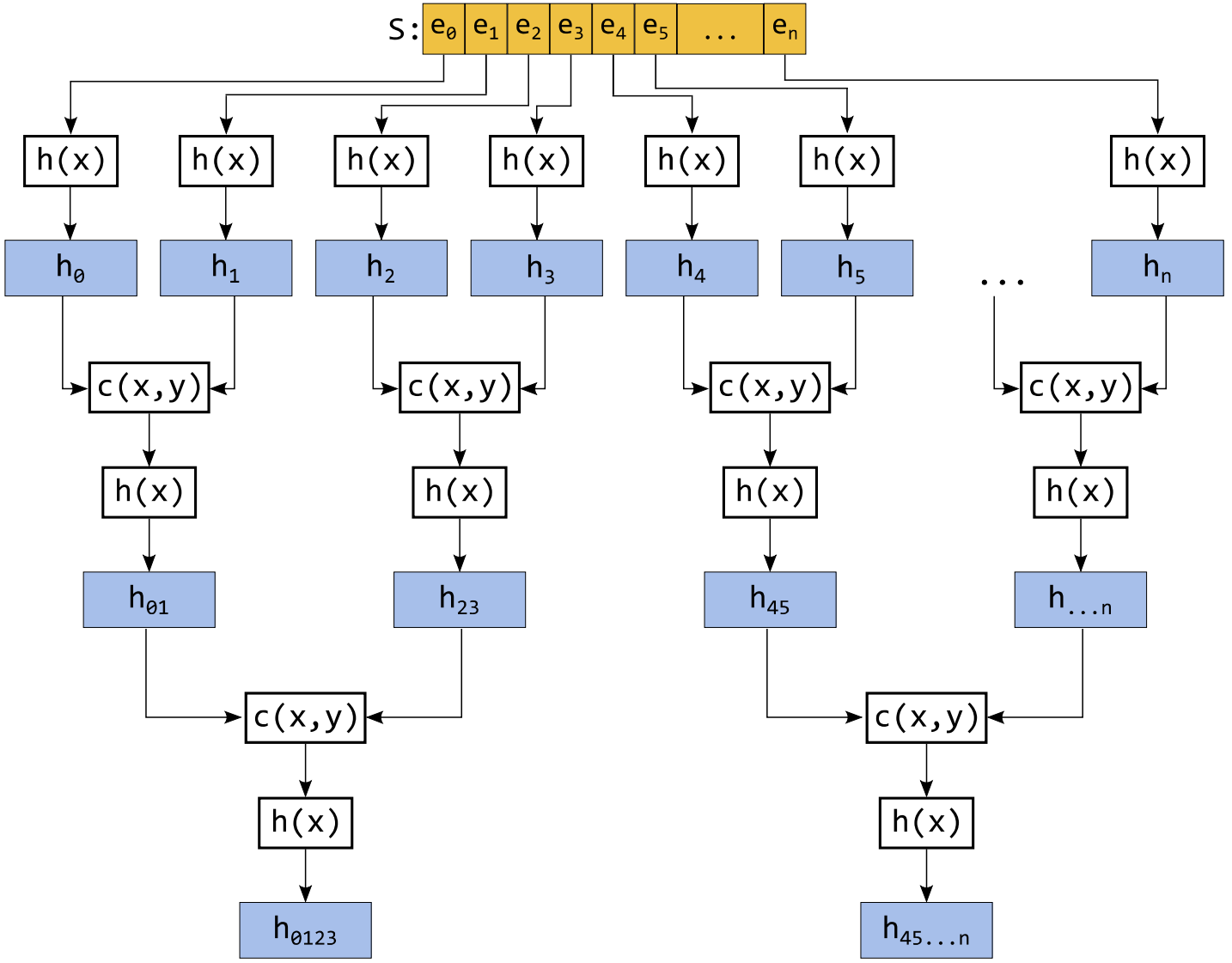

S:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4
h5

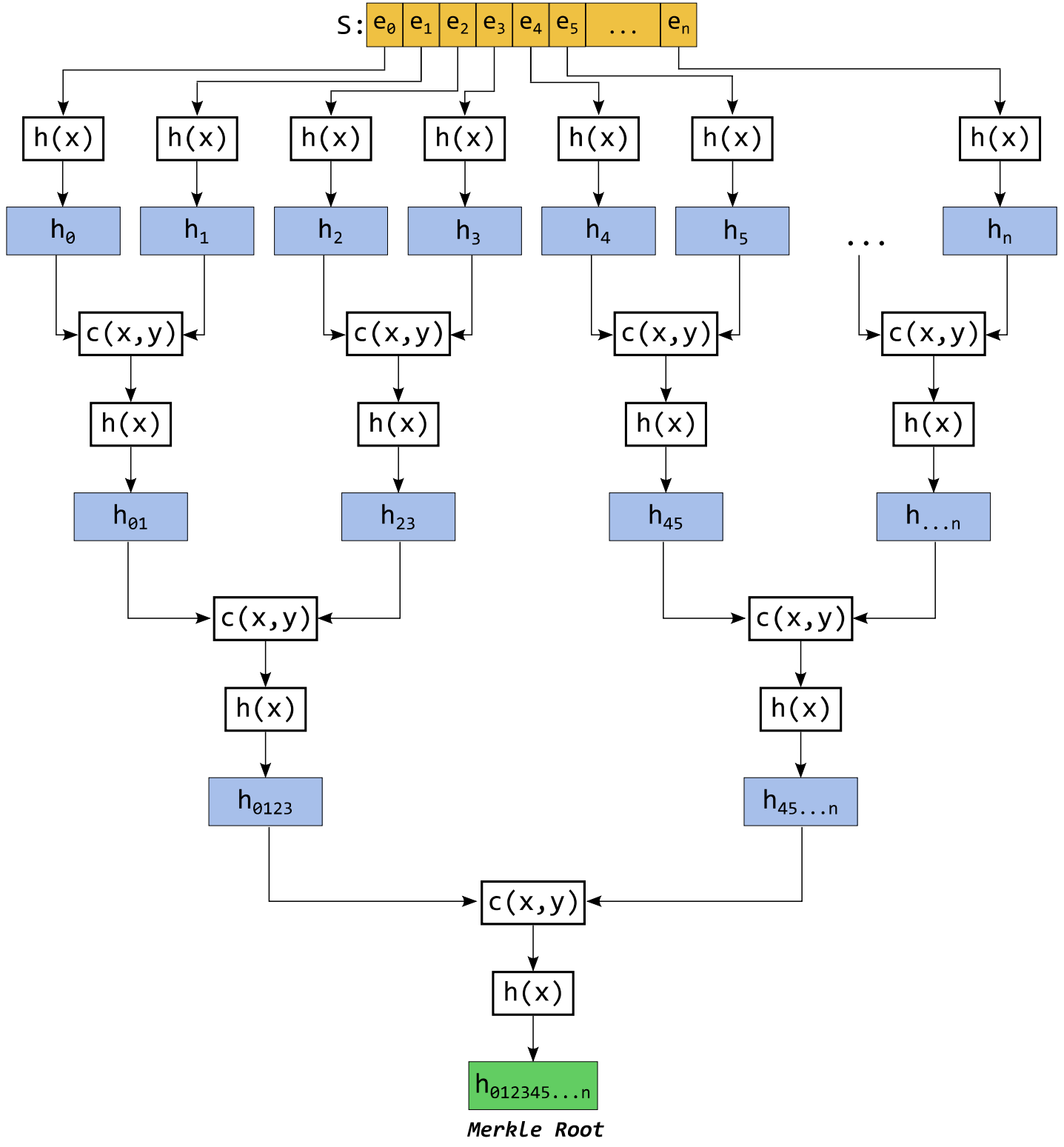

s:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4
h5
...
hn


s: e0 e1 e2 e3 e4 e5 ... en
h(x)
h0
h1
h2
h3
h4
h5
...
hn
c(x,y)
h0h1


s:
e0
e1
e2
e3
e4
e5
...
en
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4
h5
...
hn
c(x,y)
h(x)
h01

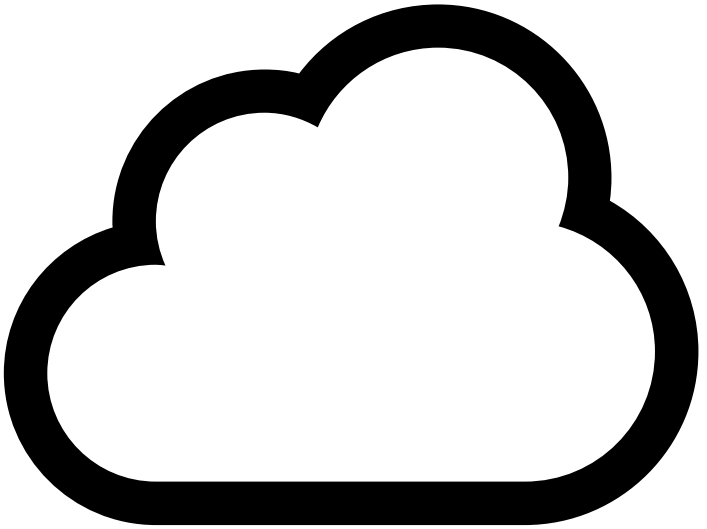

S: e0 e1 e2 e3 e4 e5 ... en
h(x)
h0
h1
h2
h3
h4
h5
...
hn
c(x,y)
h01
h23
h45
h...n


S: e0 e1 e2 e3 e4 e5 ... en
h(x)
h0 h1 h2 h3 h4 h5 ... hn
c(x,y)
h01 h23 h45 h...n
h0123 h45...n

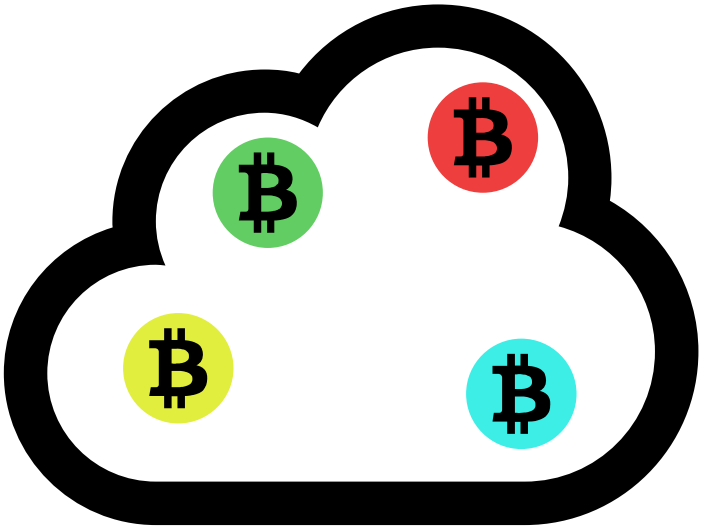

S: e0 e1 e2 e3 e4 e5 ... en
h(x)
h0 h1 h2 h3 h4 h5 ... hn
c(x,y)
h01 h23 h45 h...n
h0123 h45...n
h012345...n
Merkle Root

Bitcoin

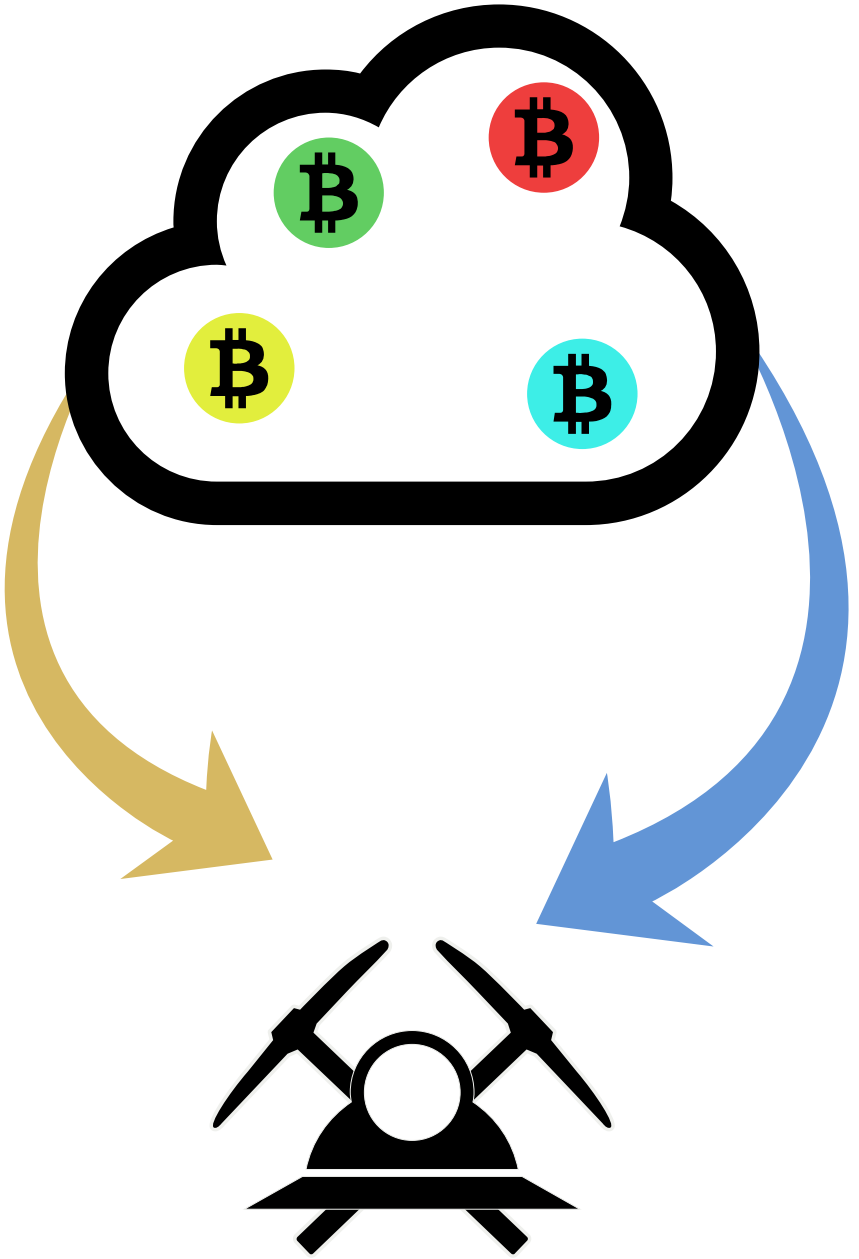
Bitcoin P2P Network



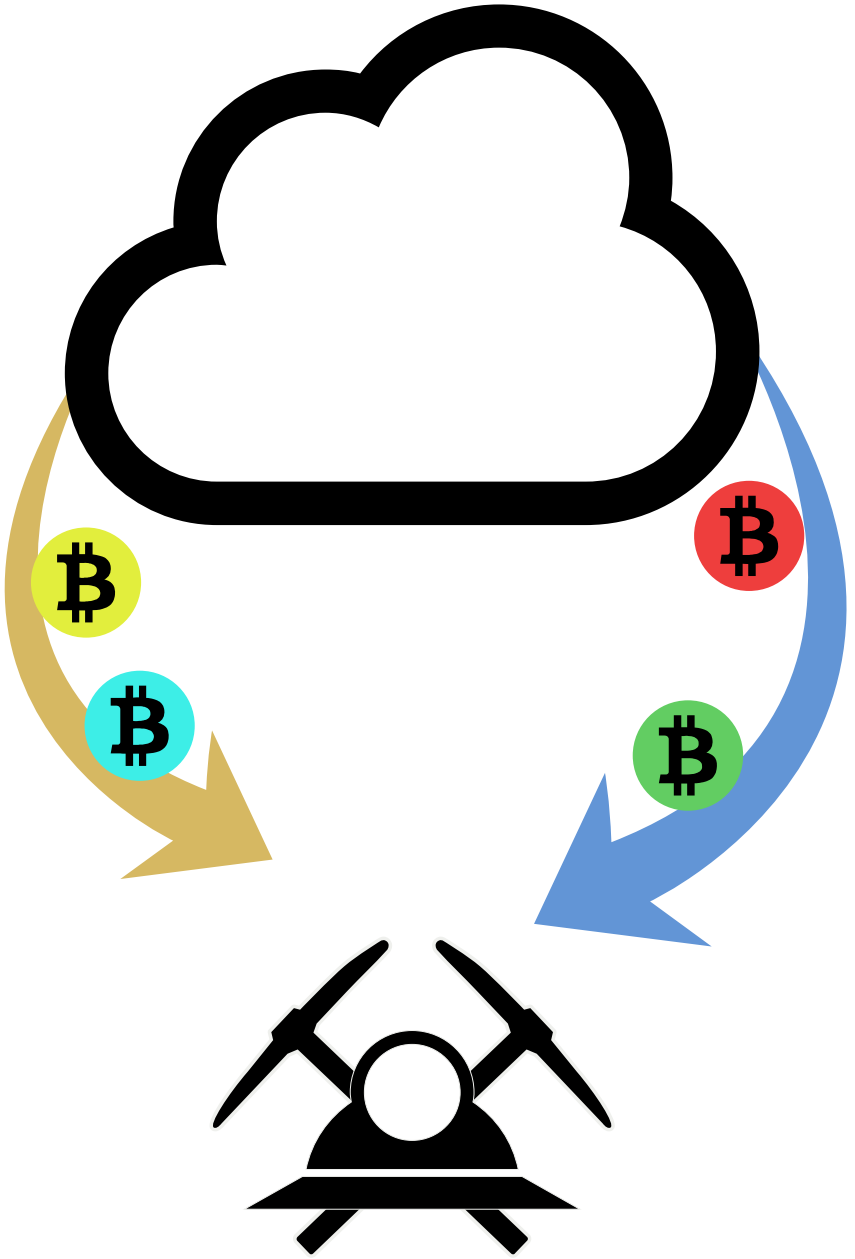
Bitcoin P2P Network



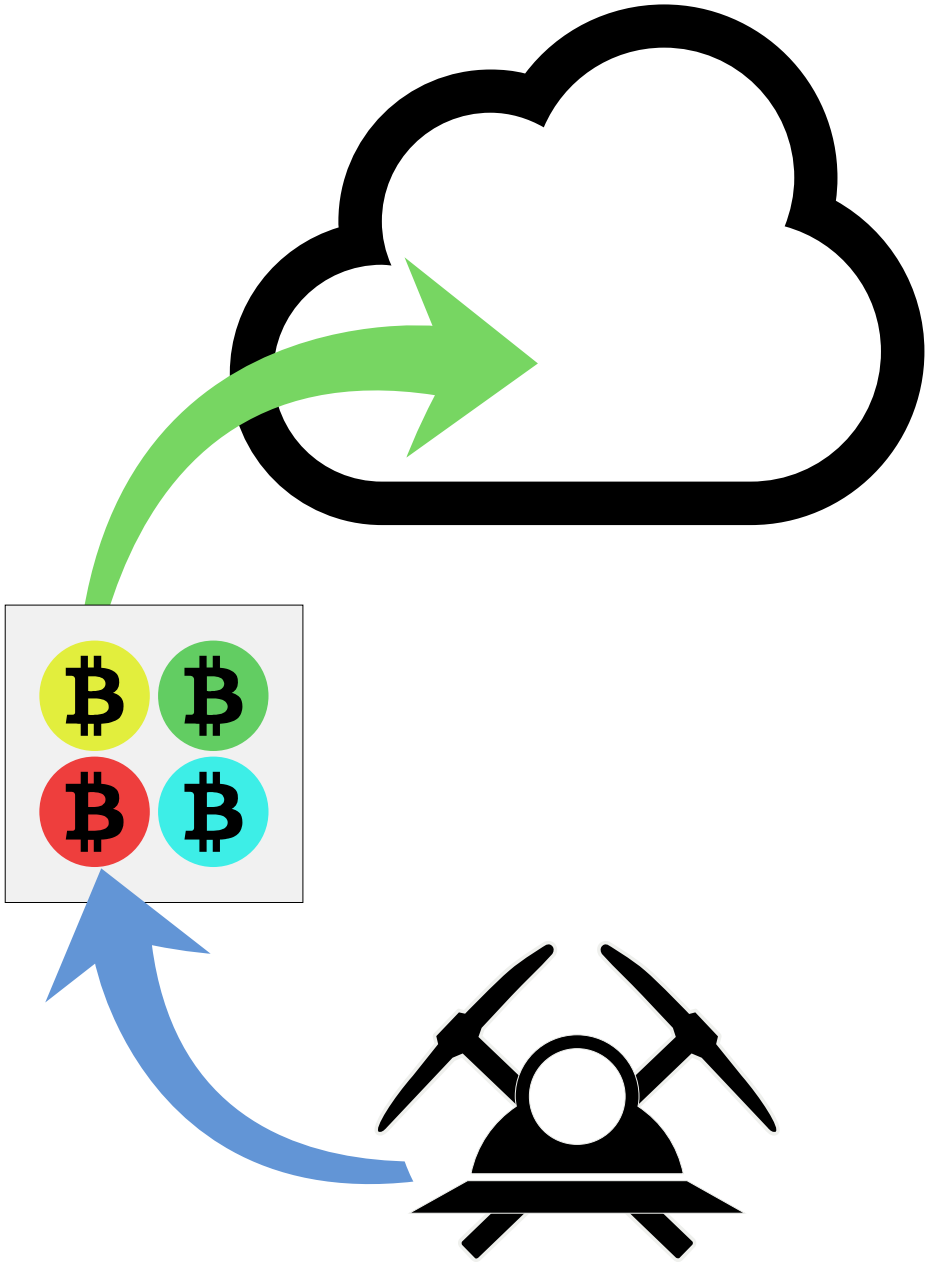
Bitcoin P2P Network



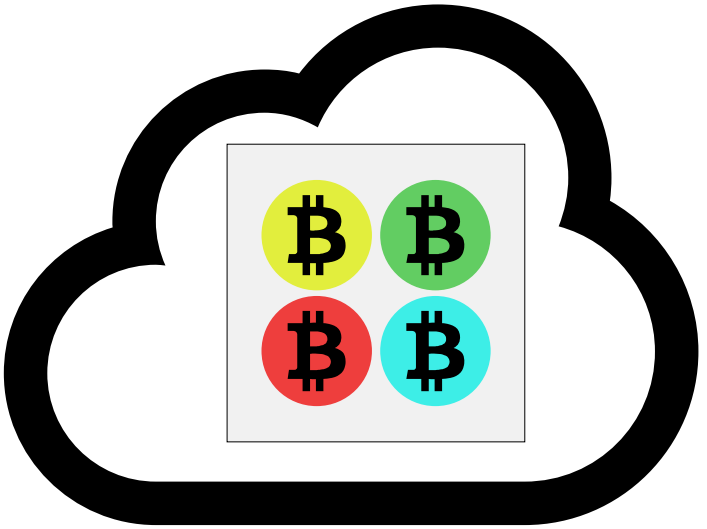
Bitcoin P2P Network



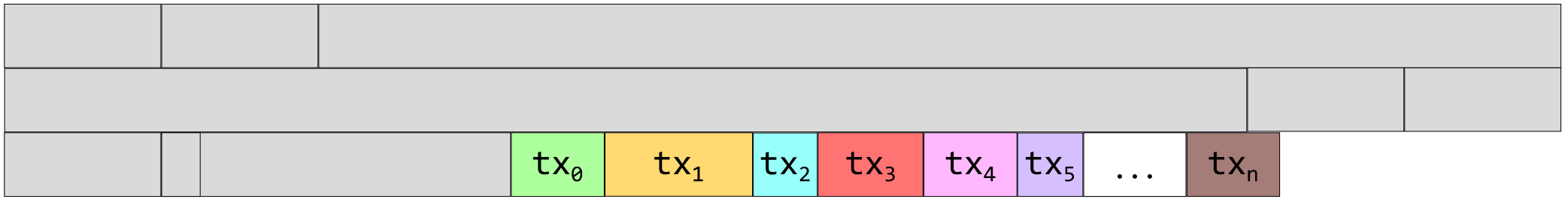
Bitcoin P2P Network



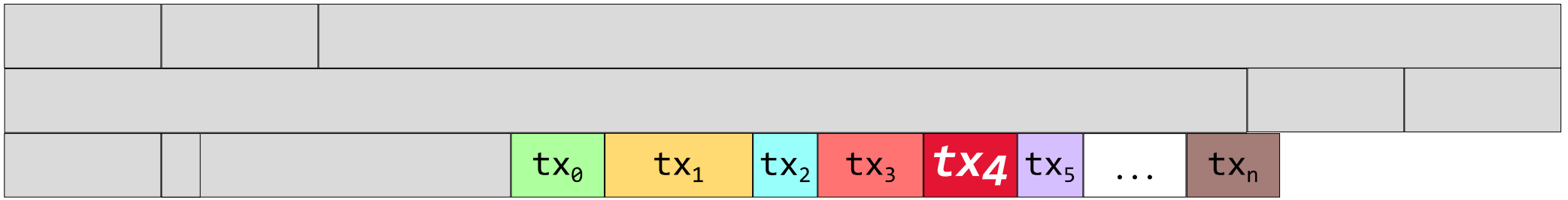
Bitcoin P2P Network



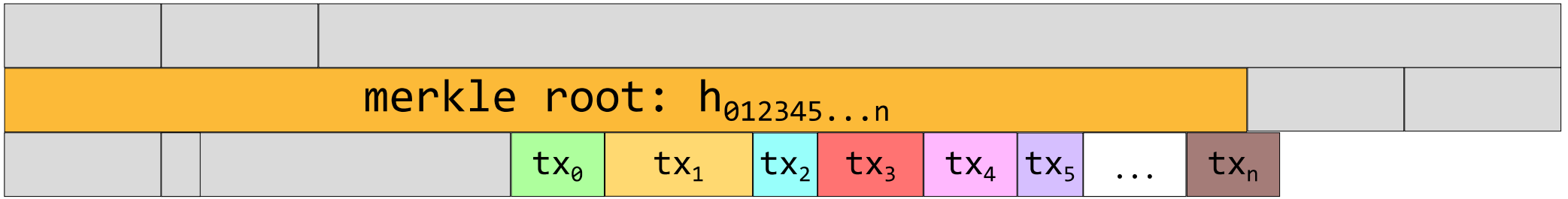
Bitcoin block structure:

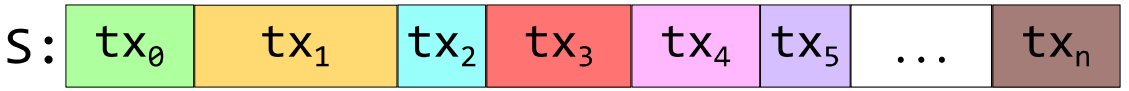


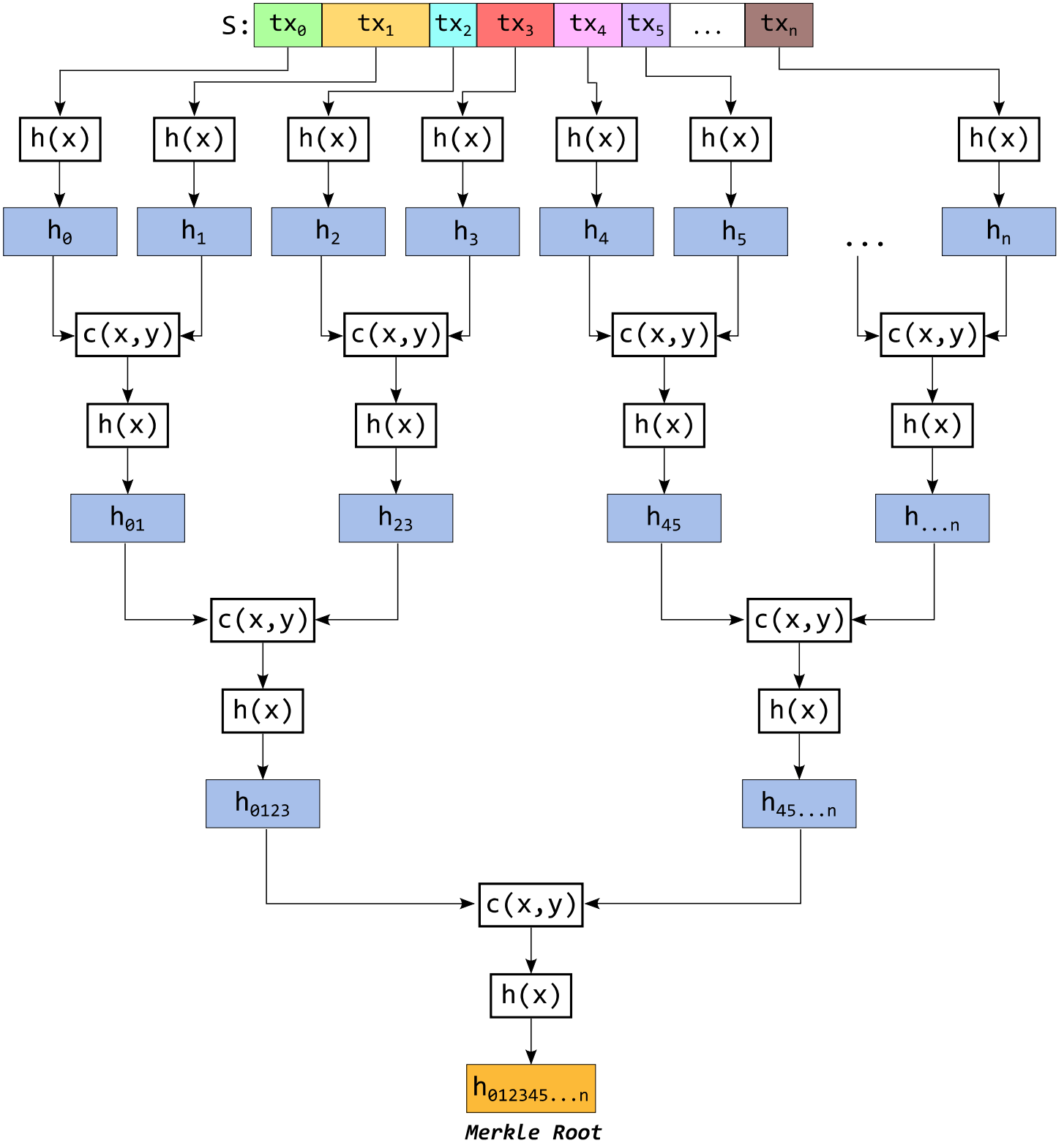
Bitcoin block structure:



Bitcoin block structure:



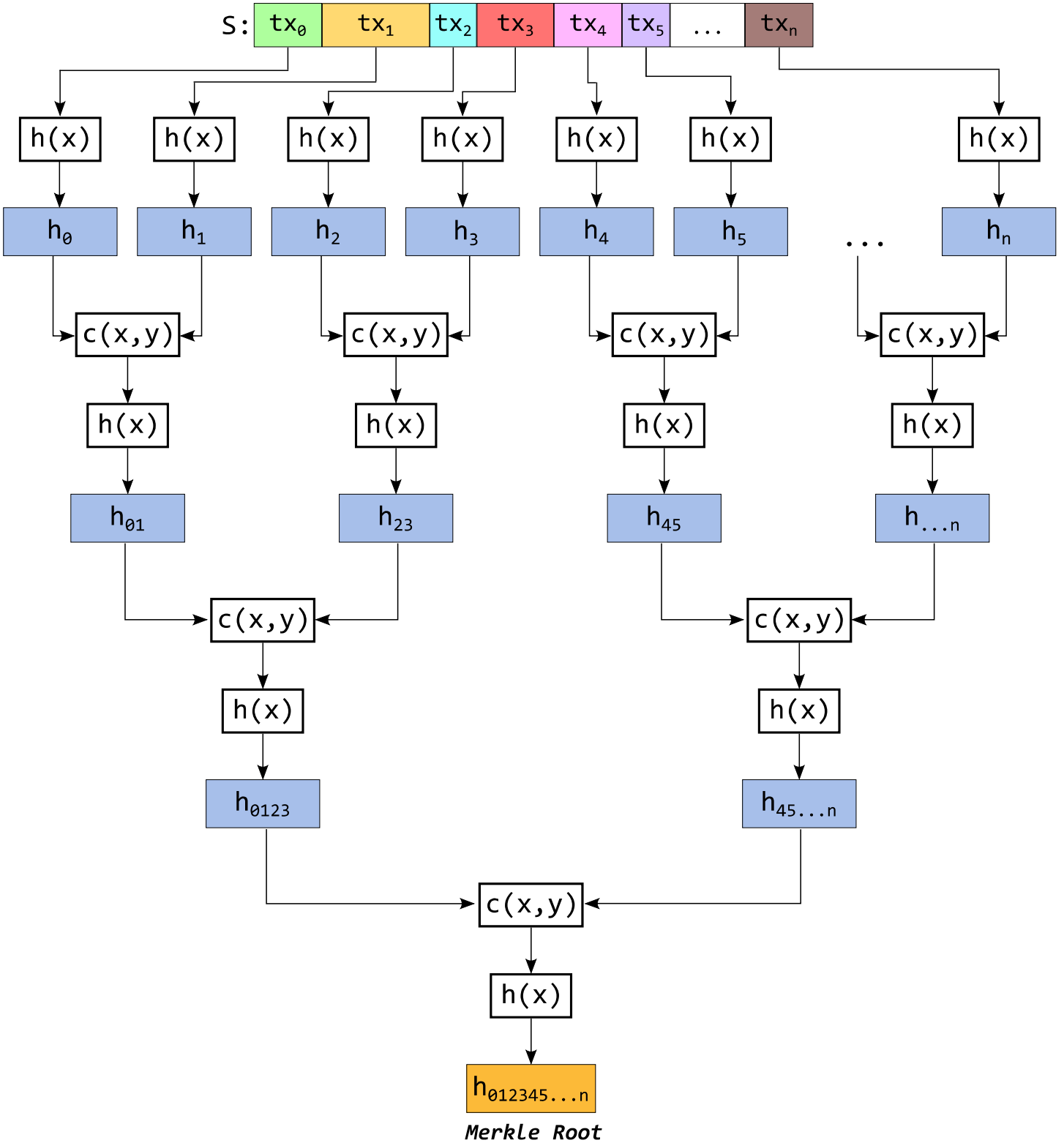

S:
tx0
tx1
tx2
tx3
tx4
tx5
...
txn


S:
tx0
tx1
tx2
tx3
tx4
tx5
...
txn
h(x)
h0
h1
h2
h3
h4
h5
...
hn
c(x,y)
h01
h23
h45
h...n
h0123
h45...n
h012345...n
Merkle Root

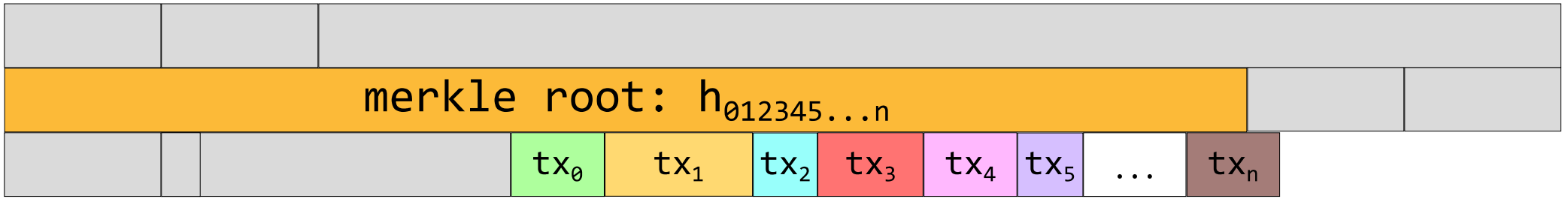
h(x) =

double sha256(x) =

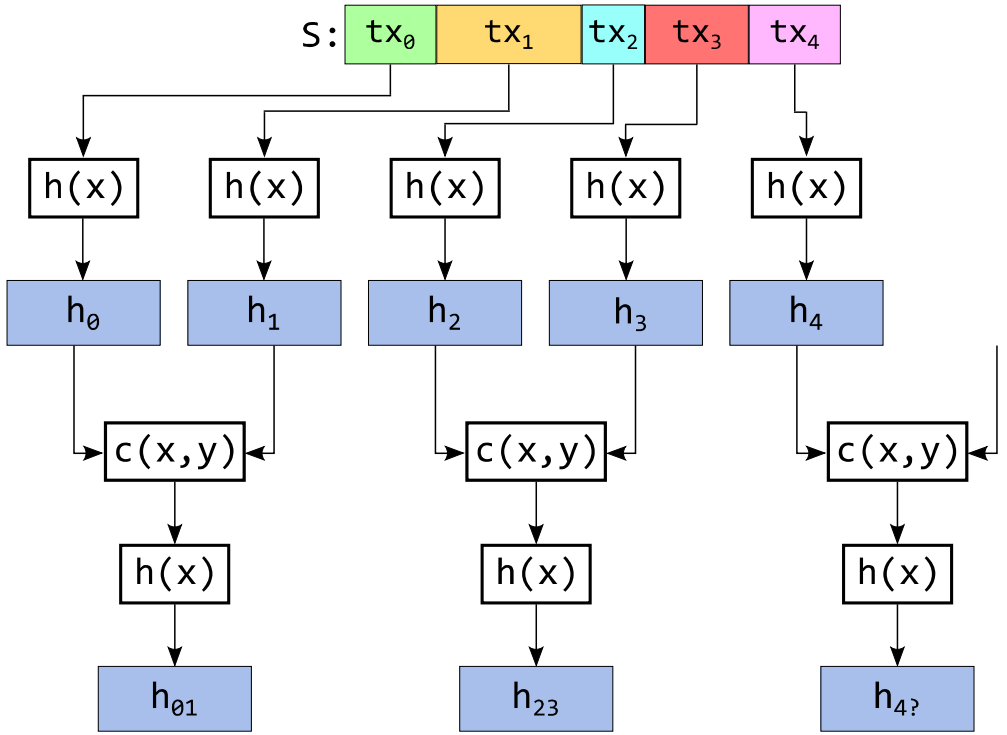
sha256(sha256(x))

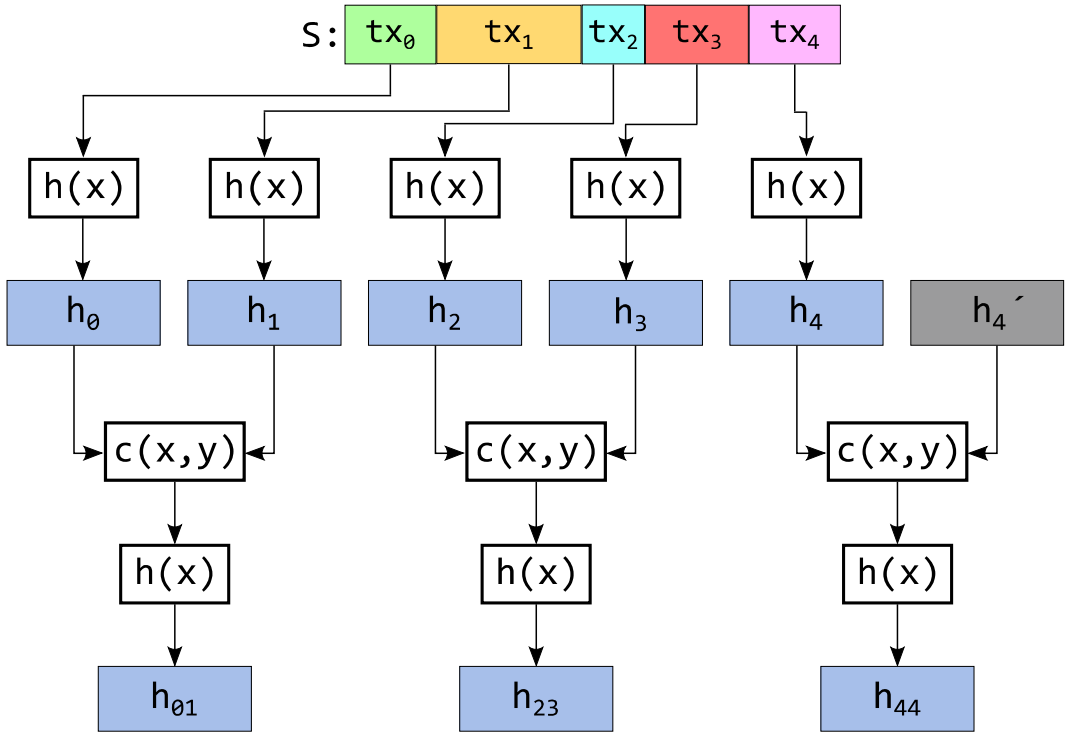

S:
tx0
tx1
tx2
tx3
tx4
tx5
...
txn
h(x)
h0
h1
h2
h3
h4
h5
...
hn
c(x,y)
h01
h23
h45
h...n
h0123
h45...n
h012345...n
Merkle Root

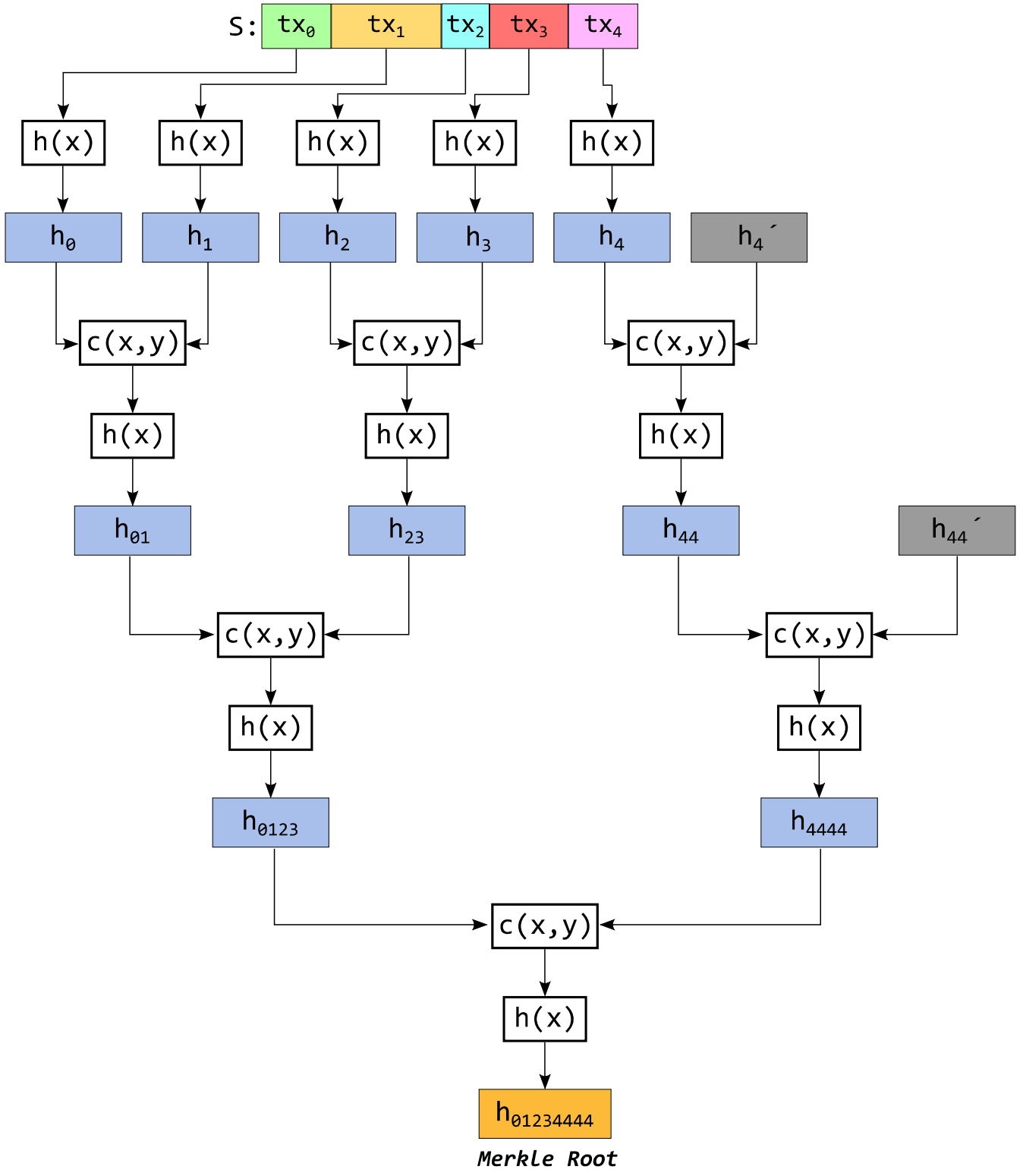
Bitcoin block structure:



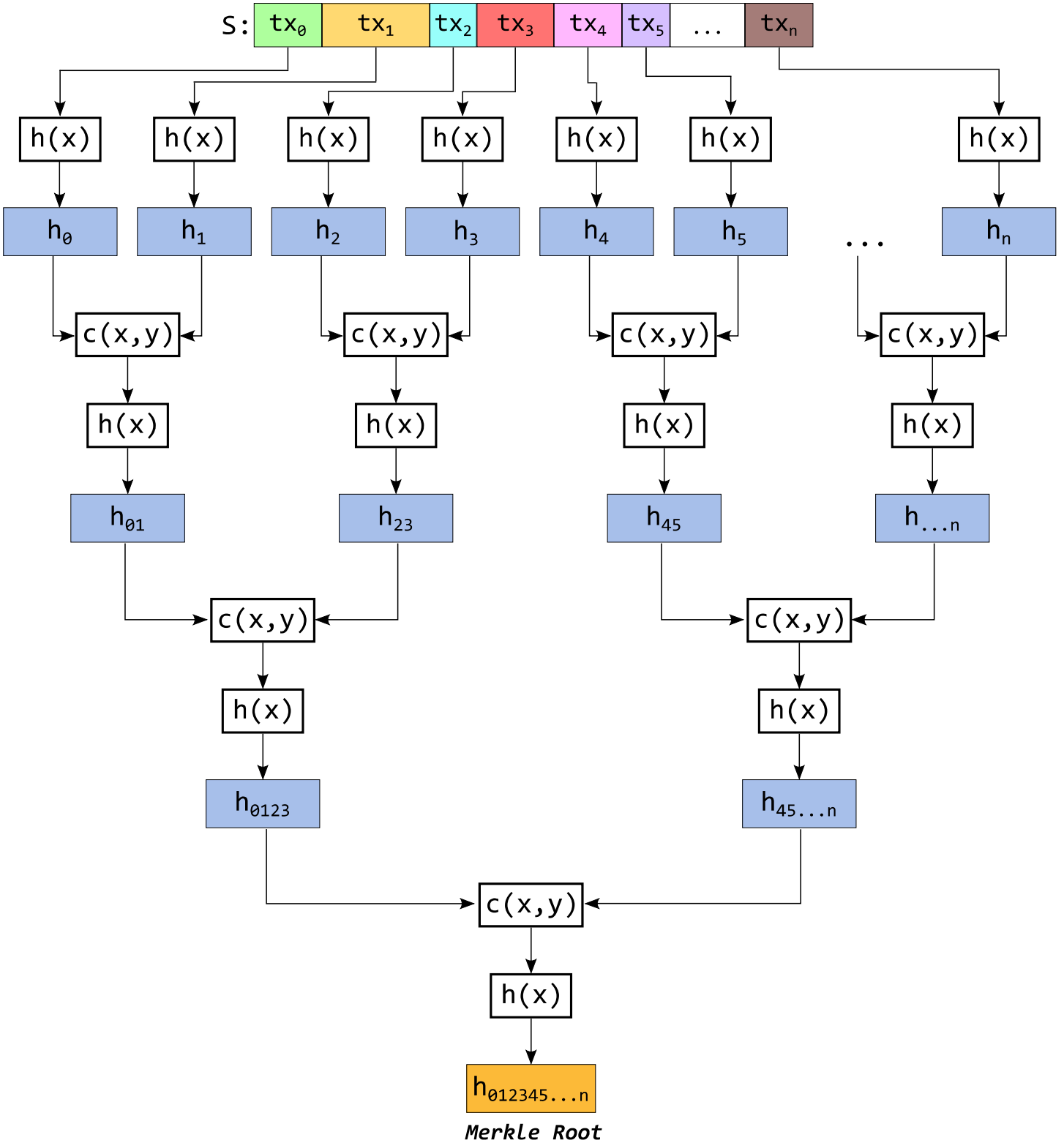

S:
tx_0
tx_1
tx_2
tx_3
tx_4


S:
tx0
tx1
tx2
tx3
tx4
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4
c(x,y)
c(x,y)
c(x,y)
h(x)
h(x)
h(x)
h01
h23
h4?


S:
tx0
tx1
tx2
tx3
tx4
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4
h4´
c(x,y)
c(x,y)
c(x,y)
h(x)
h(x)
h(x)
h01
h23
h44


S:
tx0
tx1
tx2
tx3
tx4
h(x)
h0
h1
h2
h3
h4
h4´
c(x,y)
h01
h23
h44
h44´
h0123
h4444
h01234444
Merkle Root

¿Complejidad Computacional?


S:
tx0
tx1
tx2
tx3
tx4
tx5
...
txn
h(x)
h0
h1
h2
h3
h4
h5
...
hn
c(x,y)
h01
h23
h45
h...n
h0123
h45...n
h012345...n
Merkle Root

$$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots + 1$$

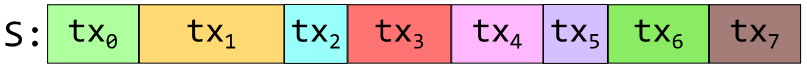
$$n = 2^k$$

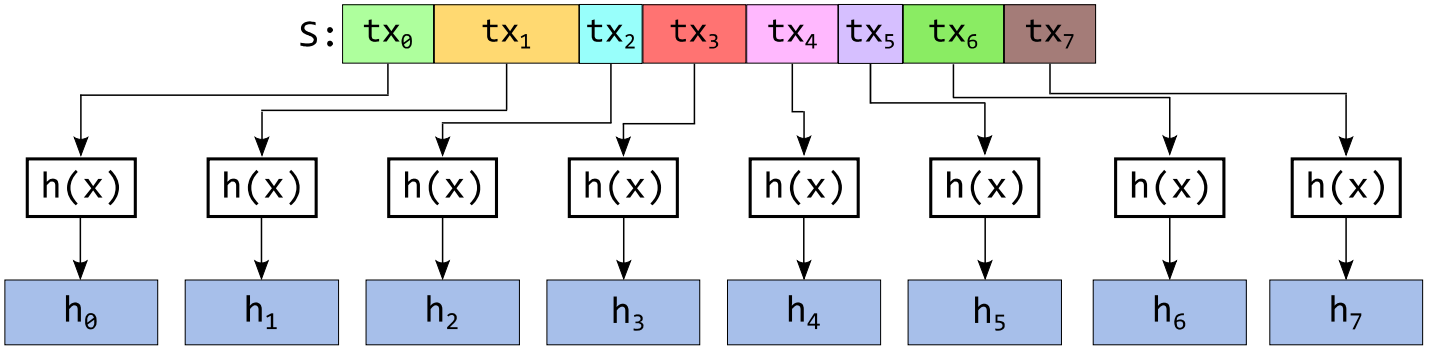
$$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + ... + 1$$

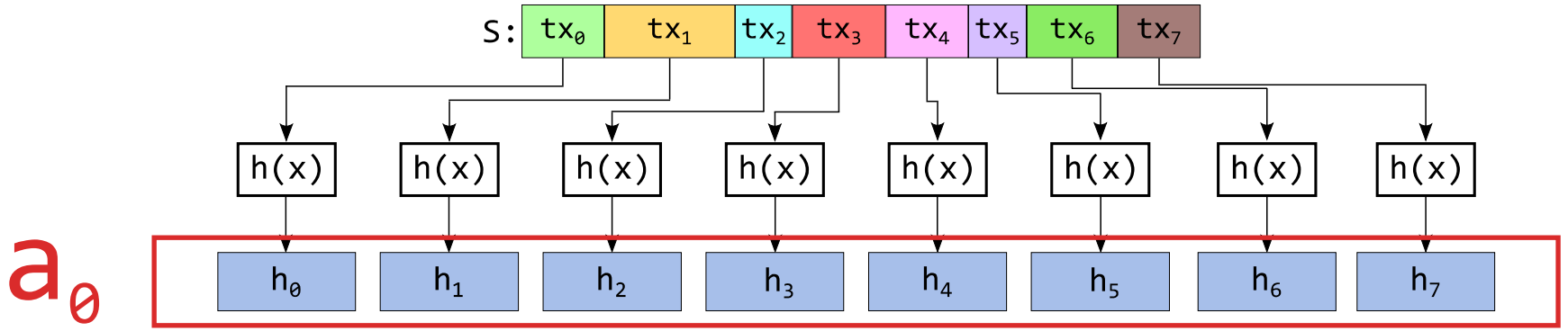
$$= \sum_{j=0}^{k} 2^{-j} n$$

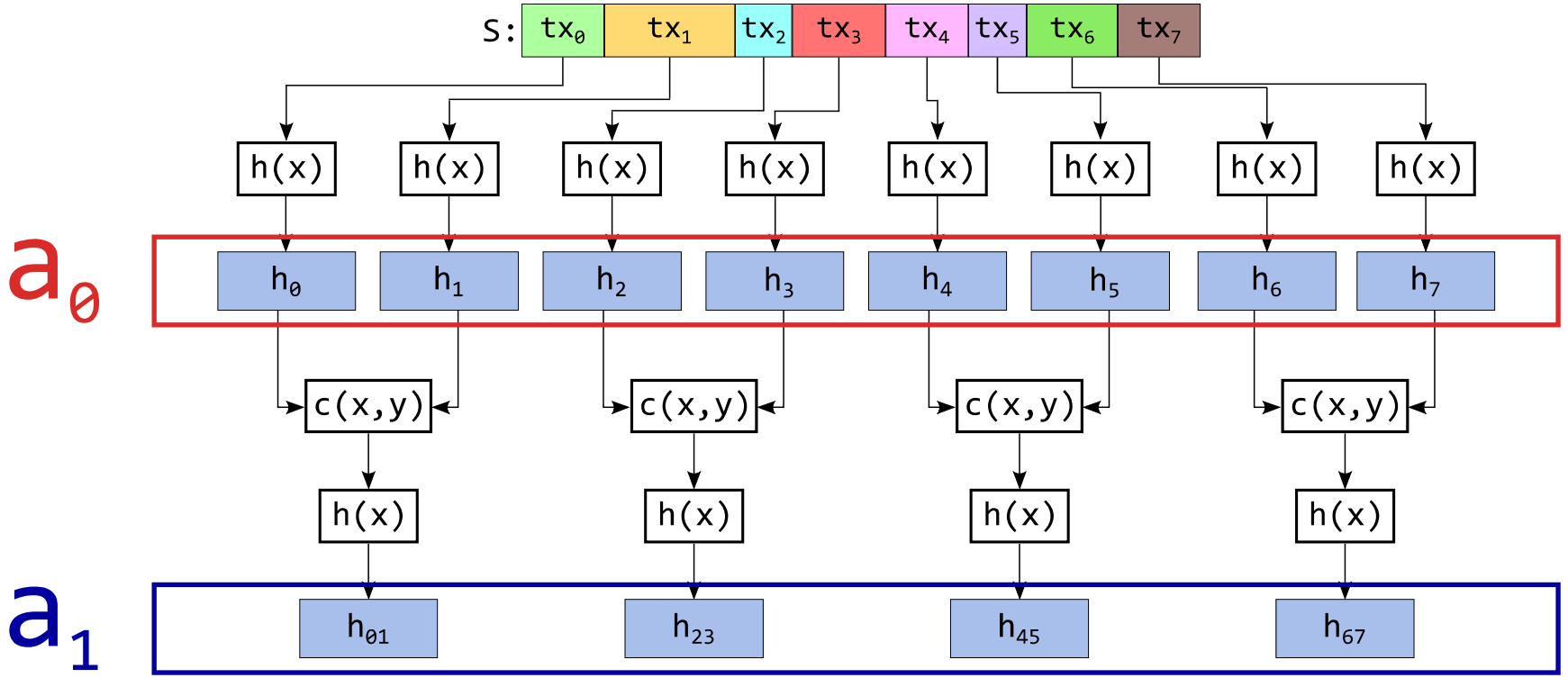
$$= 2n - 1$$

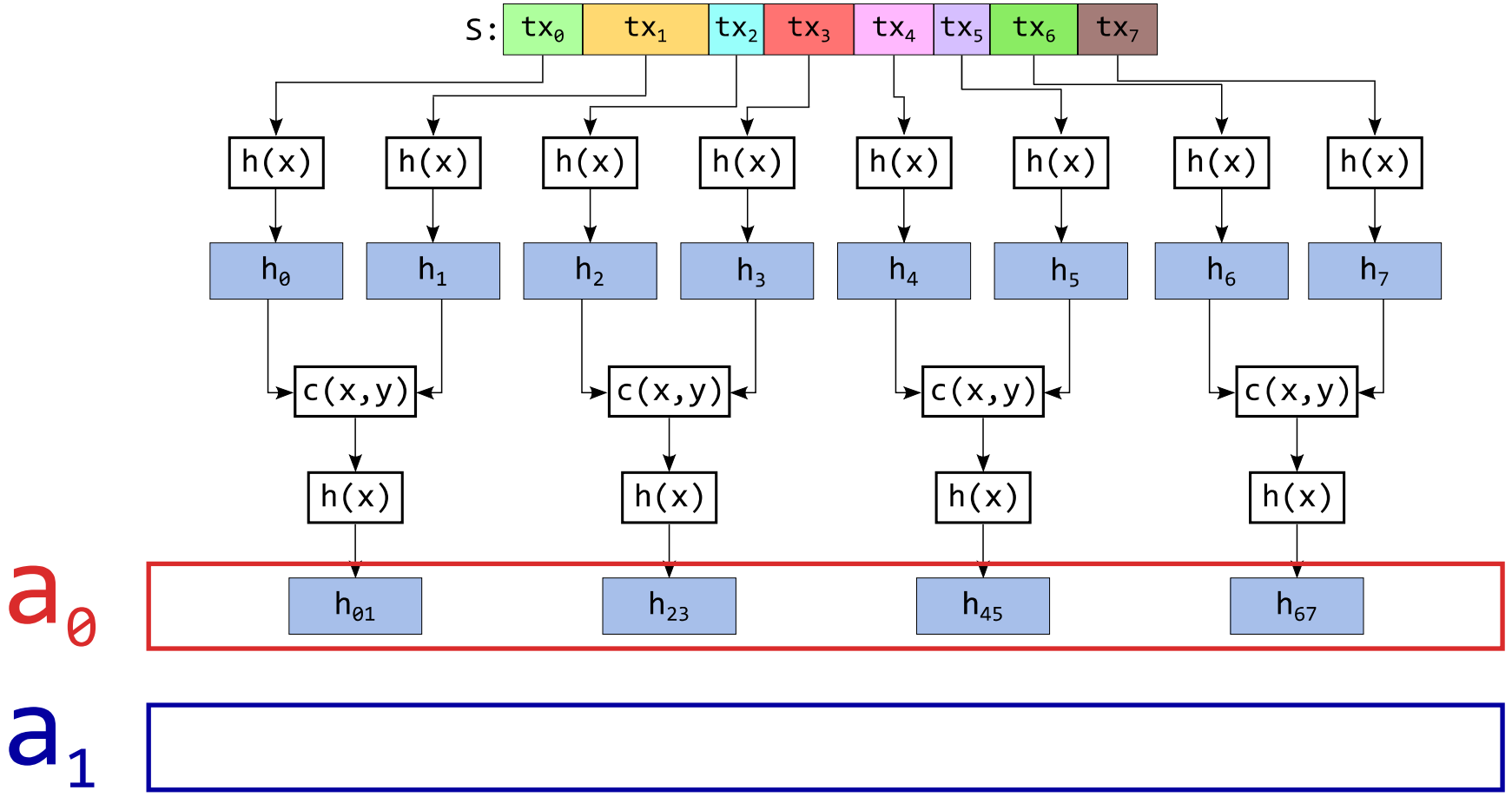
¿Cómo lo implementamos?

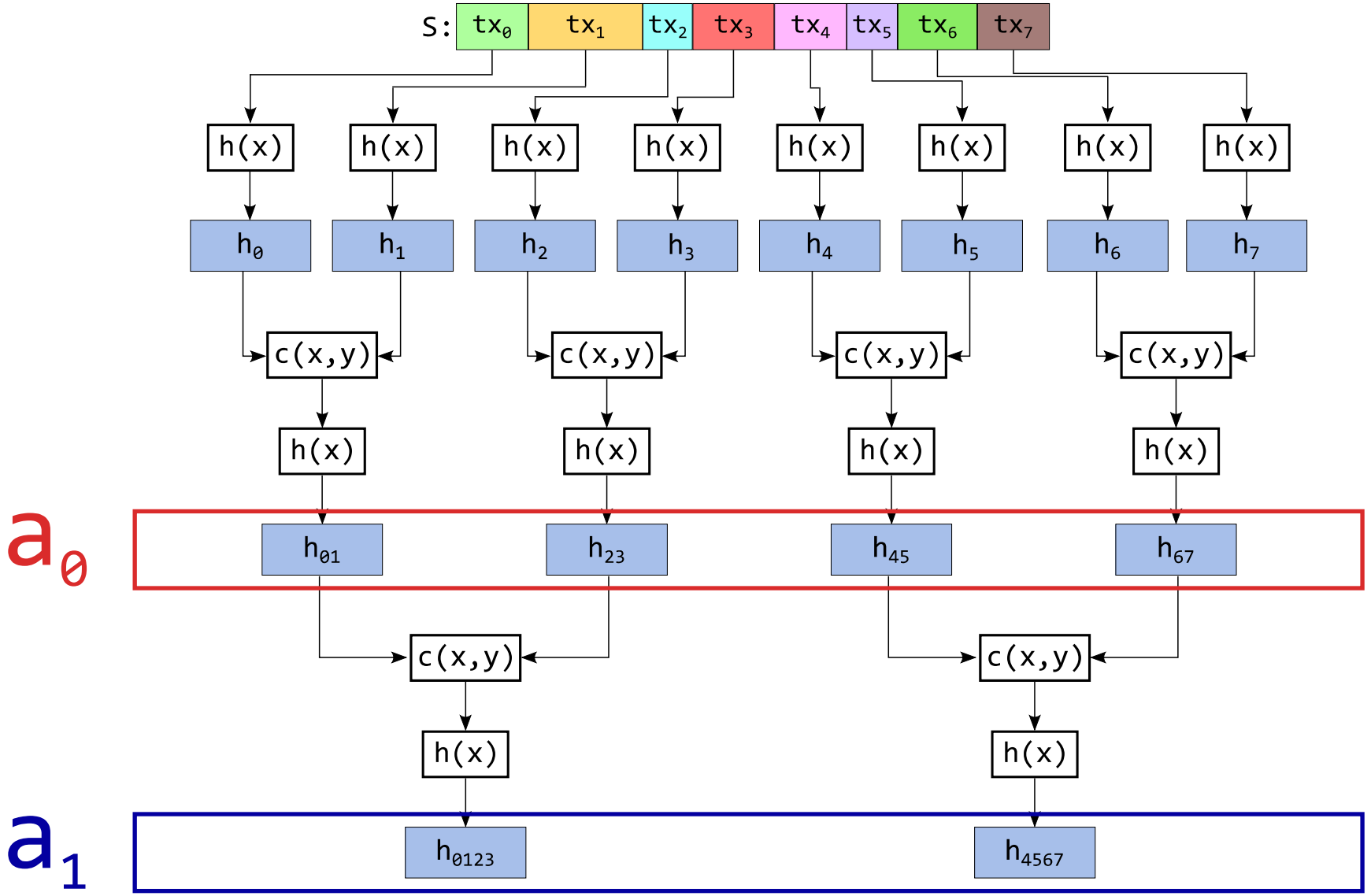

S:
tx0
tx1
tx2
tx3
tx4
tx5
tx6
tx7

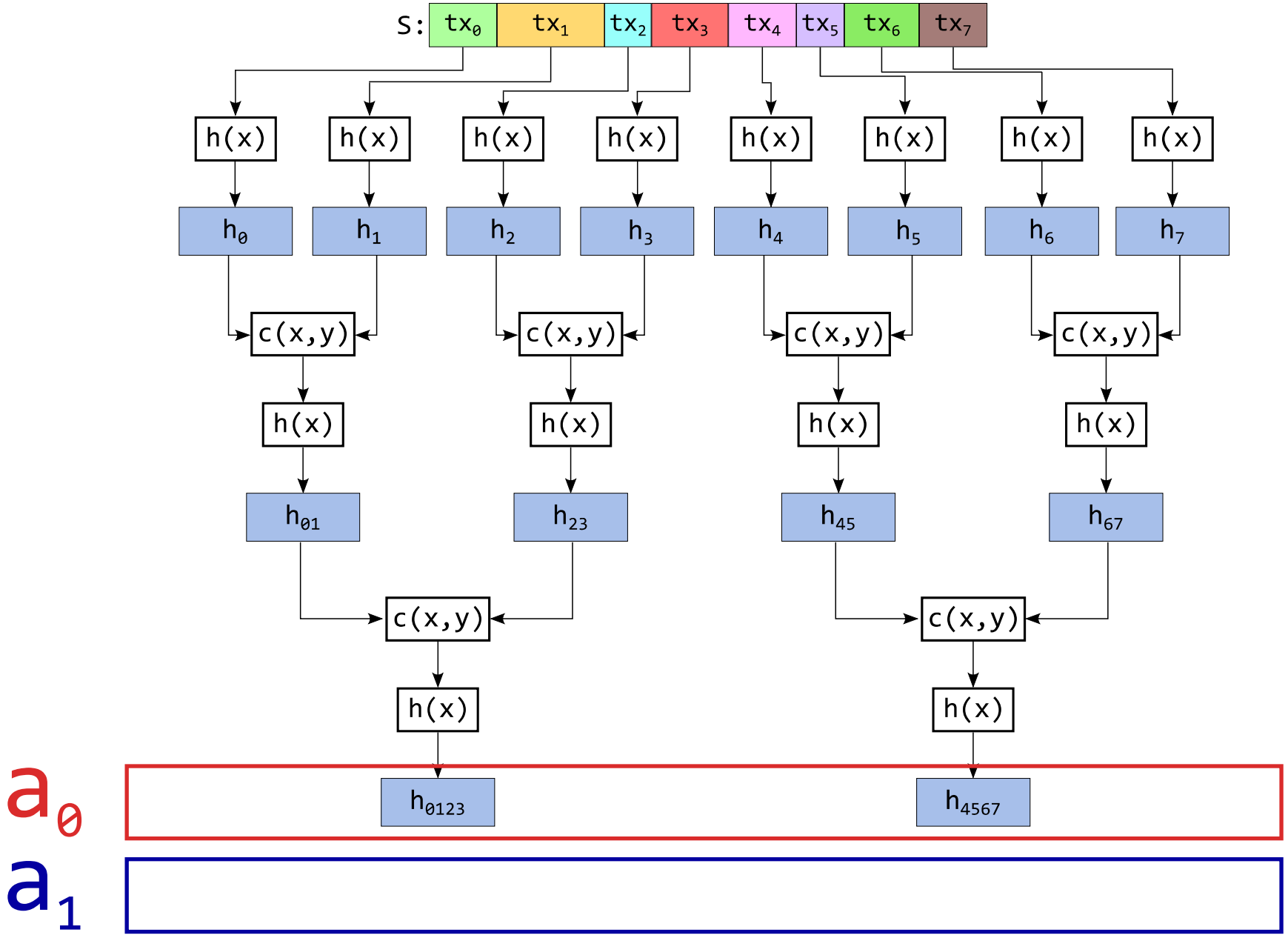

S:
tx0
tx1
tx2
tx3
tx4
tx5
tx6
tx7
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h0
h1
h2
h3
h4
h5
h6
h7

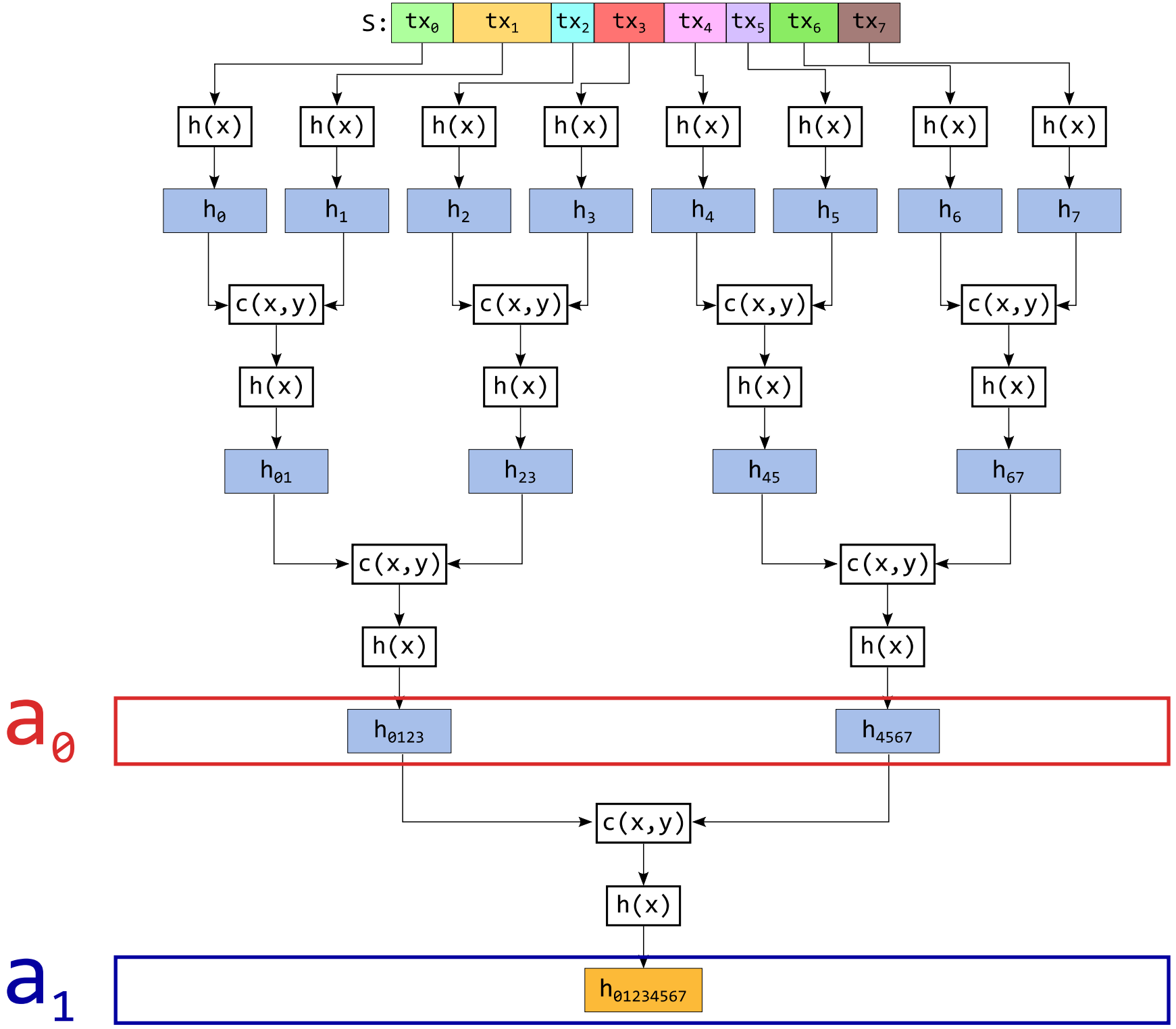

S:
tx0
tx1
tx2
tx3
tx4
tx5
tx6
tx7
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
h(x)
a0
h0
h1
h2
h3
h4
h5
h6
h7


S:
tx0
tx1
tx2
tx3
tx4
tx5
tx6
tx7
h(x)
a0
h0
h1
h2
h3
h4
h5
h6
h7
c(x,y)
h(x)
a1
h01
h23
h45
h67


S:
tx0
tx1
tx2
tx3
tx4
tx5
tx6
tx7
h(x)
h0
h1
h2
h3
h4
h5
h6
h7
c(x,y)
h01
h23
h45
h67
a0
a1


S:
tx0
tx1
tx2
tx3
tx4
tx5
tx6
tx7
h(x)
h0
h1
h2
h3
h4
h5
h6
h7
c(x,y)
h01
h23
h45
h67
h0123
h4567
a0
a1

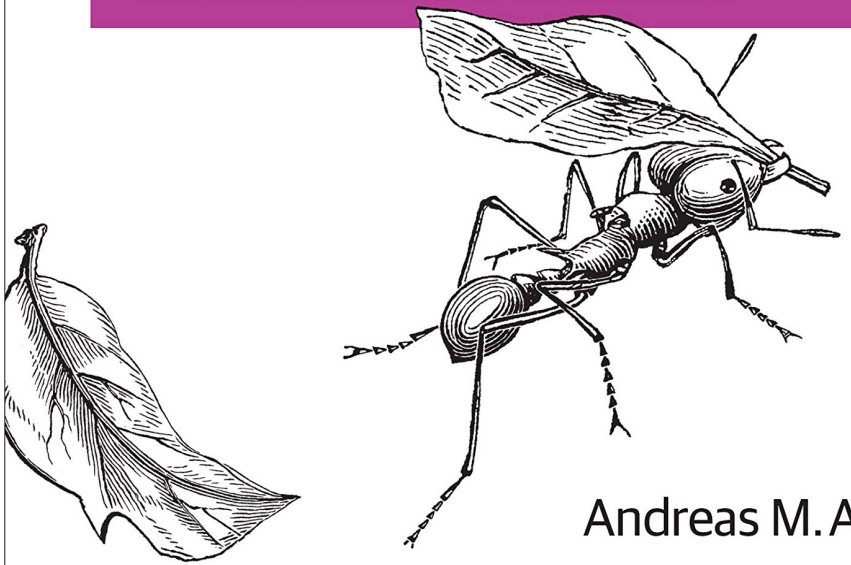

S: tx0 tx1 tx2 tx3 tx4 tx5 tx6 tx7
h(x)
h0 h1 h2 h3 h4 h5 h6 h7
c(x,y)
h01 h23 h45 h67
h0123 h4567
a0
a1


S: tx0 tx1 tx2 tx3 tx4 tx5 tx6 tx7
h(x)
h0 h1 h2 h3 h4 h5 h6 h7
c(x,y)
h01 h23 h45 h67
h0123 h4567
h01234567
a0
a1

O'REILLY®

Mastering Bitcoin

UNLOCKING DIGITAL CRYPTOCURRENCIES



Andreas M. Antonopoulos

Example 7-1. Building a merkle tree

```
#include <bitcoin/bitcoin.hpp>

bc::hash_digest create_merkle(bc::hash_digest_list& merkle)
{
    // Stop if hash list is empty.
    if (merkle.empty())
        return bc::null_hash;
    else if (merkle.size() == 1)
        return merkle[0];

    // While there is more than 1 hash in the list, keep looping...
    while (merkle.size() > 1)
    {
        // If number of hashes is odd, duplicate last hash in the list.
        if (merkle.size() % 2 != 0)
            merkle.push_back(merkle.back());
        // List size is now even.
        assert(merkle.size() % 2 == 0);

        // New hash list.
        bc::hash_digest_list new_merkle;
        // Loop through hashes 2 at a time.
        for (auto it = merkle.begin(); it != merkle.end(); it += 2)
        {
            // Join both current hashes together (concatenate).
            bc::data_chunk concat_data(bc::hash_size * 2);
            auto concat = bc::make_serializer(concat_data.begin());
            concat.write_hash(*it);
            concat.write_hash(*(it + 1));
```

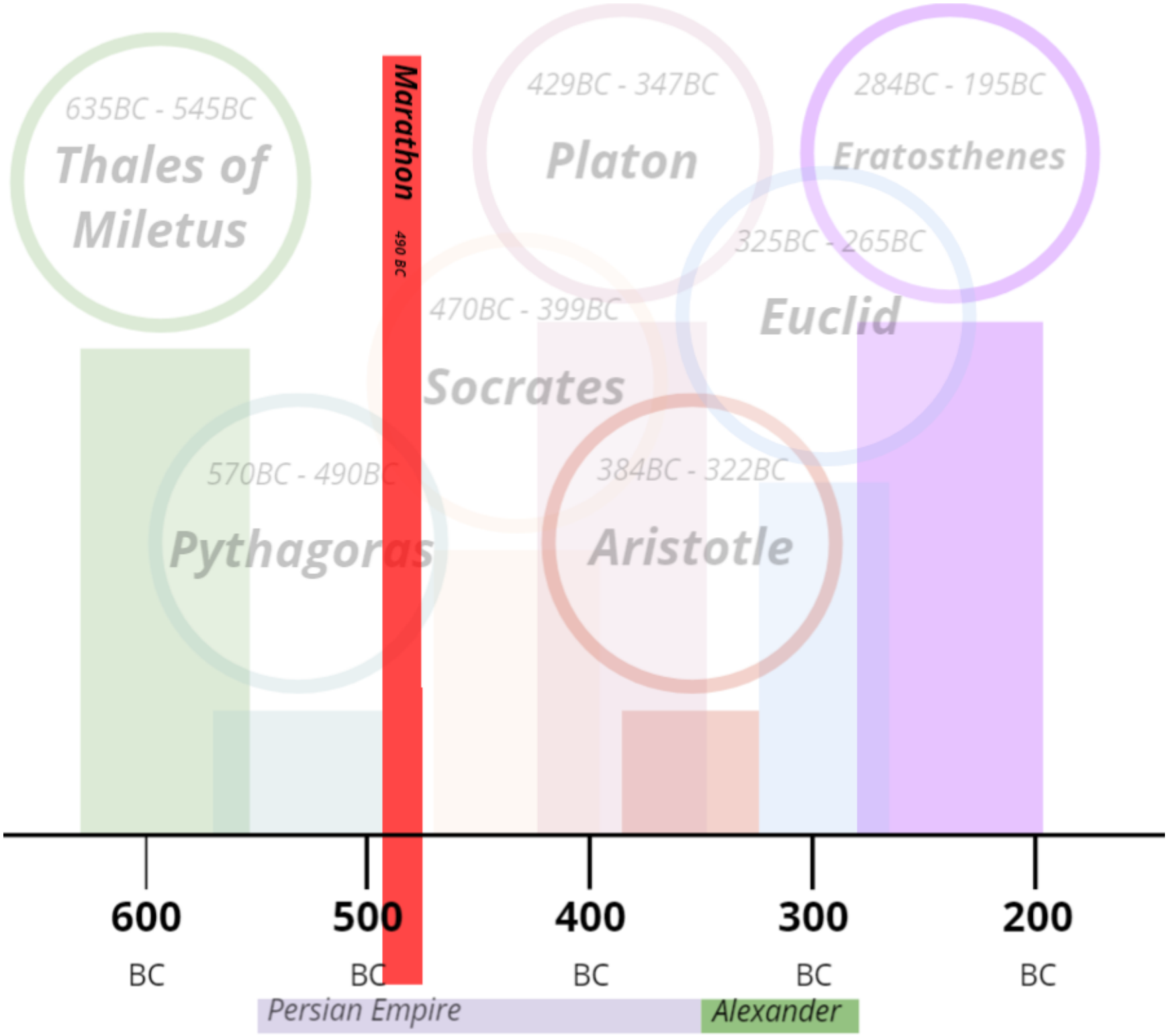
```
            assert(concat.iterator() == concat_data.end());
            // Hash both of the hashes.
            bc::hash_digest new_root = bc::bitcoin_hash(concat_data);
            // Add this to the new list.
            new_merkle.push_back(new_root);
        }
        // This is the new list.
        merkle = new_merkle;

        // DEBUG output ------------------------------------
        std::cout << "Current merkle hash list:" << std::endl;
        for (const auto& hash: merkle)
            std::cout << "  " << bc::encode_hex(hash) << std::endl;
        std::cout << std::endl;
        // -------------------------------------------------
    }
    // Finally we end up with a single item.
    return merkle[0];
}
```

https://github.com/libbitcoin/libbitcoin/blob/version3/src/chain/block.cpp#L551

¿Podemos mejorarlo?

“Scuola di Atene”, Raffaello Sanzio, ~1509-1511

- Batalla de Maratón: 490 a. C.
- Batalla de Salamina: 480 a. C.
- Batalla de Platea: 479 a. C.


635BC - 545BC
Thales of Miletus
Marathon
490 BC
429BC - 347BC
Platon
284BC - 195BC
Eratosthenes
325BC - 265BC
Euclid
470BC - 399BC
Socrates
570BC - 490BC
Pythagoras
384BC - 322BC
Aristotle
600
BC
500
BC
400
BC
300
BC
200
BC
Persian Empire
Alexander


Cándido
Mesonero Mayor de Castilla

“In summo apud illos honore Geometria fuit, itaque nihil Mathematicis illustrius. At nos metiendi ratiocinandique utilitate hujus artis terminavimus modum.”

Cicero, Tusculanarum quaestionum, Lib. I. in princ.

“Para los griegos la geometría era el mayor honor, por lo tanto nadie era más honorable que los matemáticos. Pero nosotros (los Romanos) hemos limitado la utilidad de este arte a medir y calcular.”

Matemática en otras civilizaciones

मूळांतरभूमिवर्गो वंशोद्धृतस्तेन पृथग्यु[hole]नौः॥ वंशस्तदर्धे भवतः क्रमेण
वंशस्य खंडे श्रुतिकोटिरूपे॥७६॥ अत्रोद्दे[hole]:॥ यदि समभुवि वेणुर्द्वित्रि
पाणिप्रमाणो गणक पवनवेगादेकदेशे स भग्नः॥ भुवि नृपमित १६ ह
स्तेष्वेव लग्नं तदग्रं कथय कतिषु मूलादेष भग्नः करेषु॥७०॥ न्यासः॥ आते
ऊर्ध्वाधः खंडे २०।१२ बाहुकर्णयोर्योगे

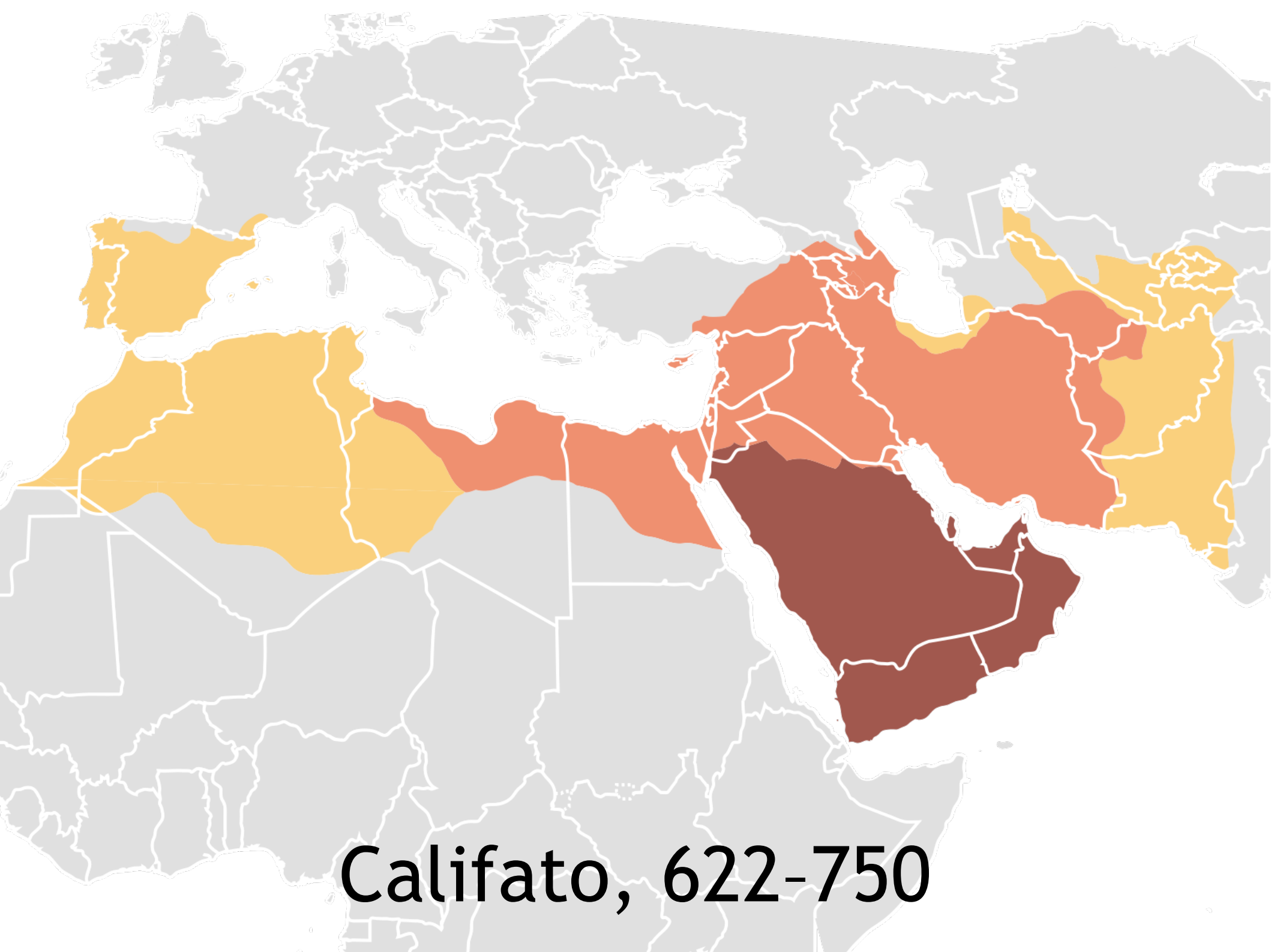
कोटौ च दृष्टे करसाद्
त्रं॥ स्तंभस्य वर्गो हि बिलांतरेण भक्तः फलं व्यालबिलां
तरालात्॥ शोध्यं तदर्धप्रमितैः करैः स्याद्बिलायतो व्याल
कलापियोगः॥७७॥ अत्रोद्देशकः॥ अस्ति स्तंभतले बिलं
तदुपरि क्रीडाशिखंडी स्थितः स्तंभे हस्तनवोच्छ्रिते त्रिगुणितस्तंभप्रमाणां
तरे॥ दृष्ट्वाहिं बिलमाव्रजंतमपतत्तिर्यक् स तस्योपरि क्षिप्रं ब्रूहि तयोर्बि
लात्कतिमितैः साम्येन गत्योर्युतिः॥७८॥ लब्धाबिलांतरे कतिहस्ताः १२।१५

Sistemas de numeración:

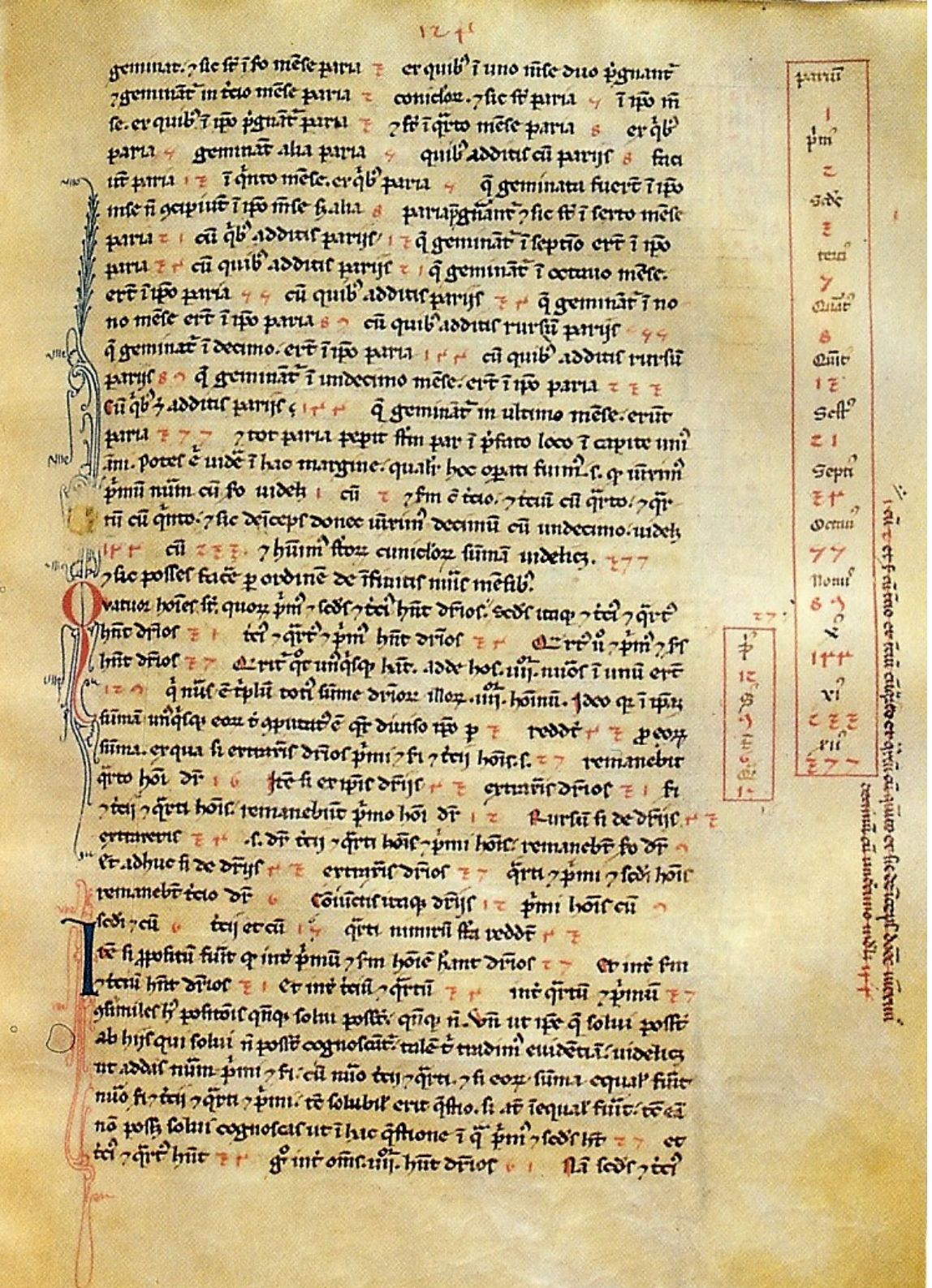
- No posicionales: ej. Egipcio, Romano.

- Posicionales: ej. Babilónico, Indo-arábigo.

Notación posicional


Califato, 622-750

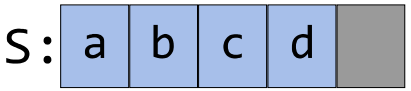
Leonardo Pisano, 1170-1240

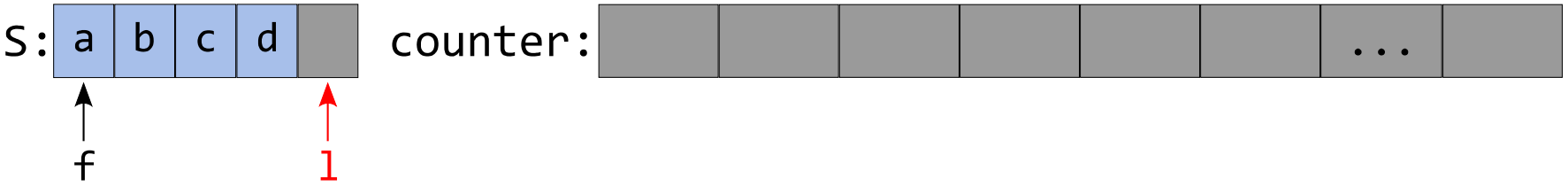


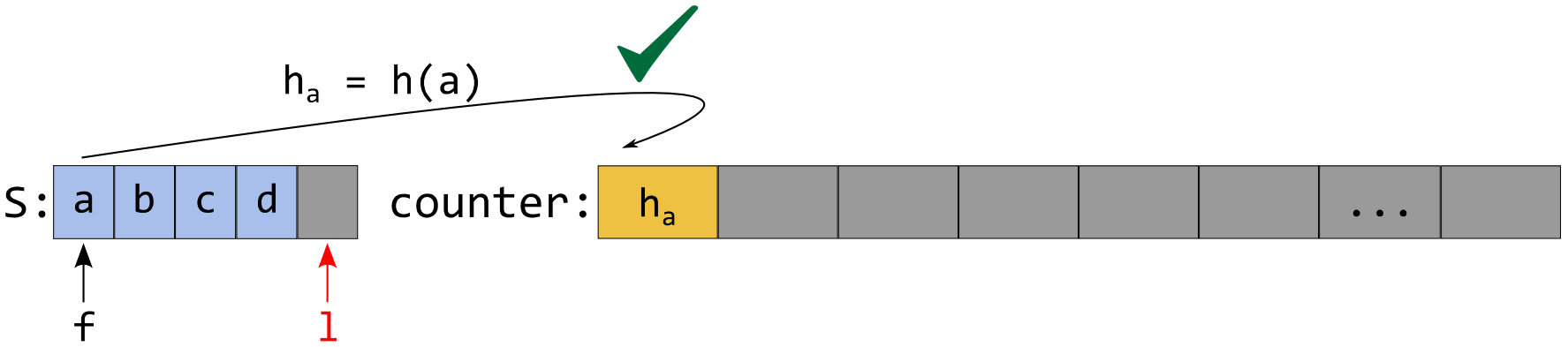
“Liber Abaci”, 1203.

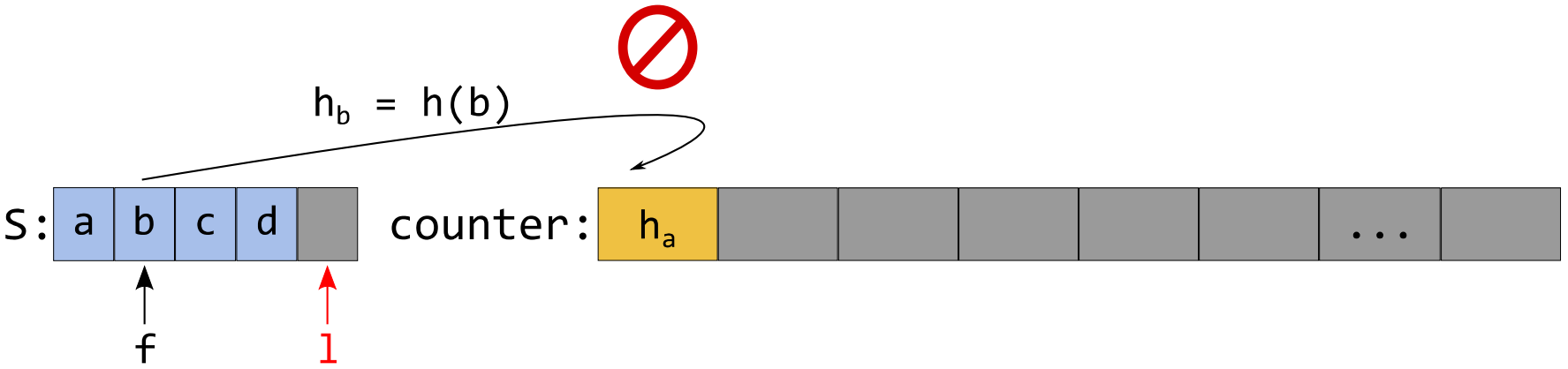
“El Libro de Cálculo”

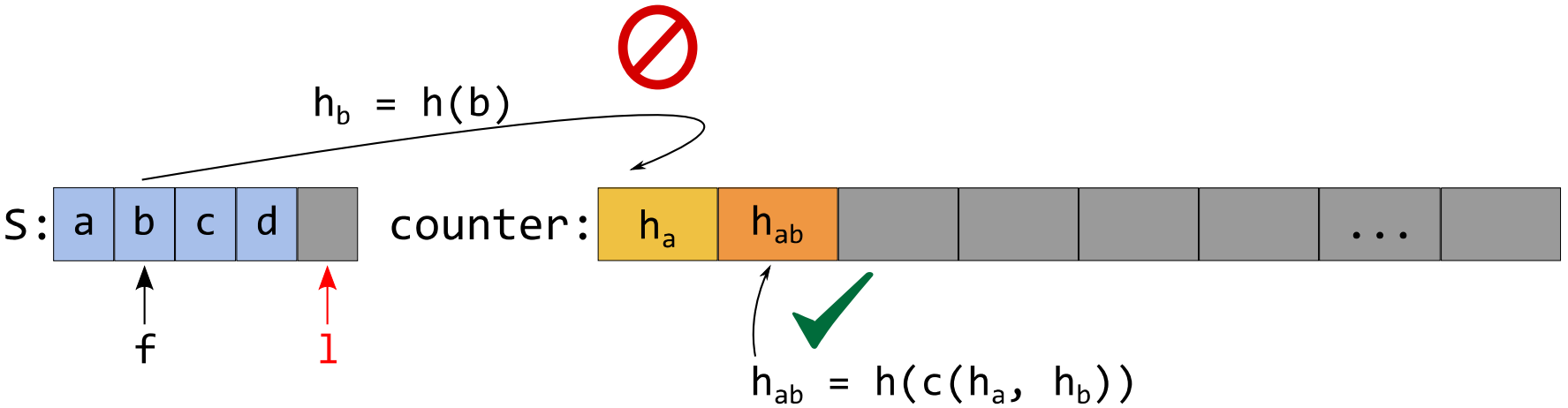
Contador Binario

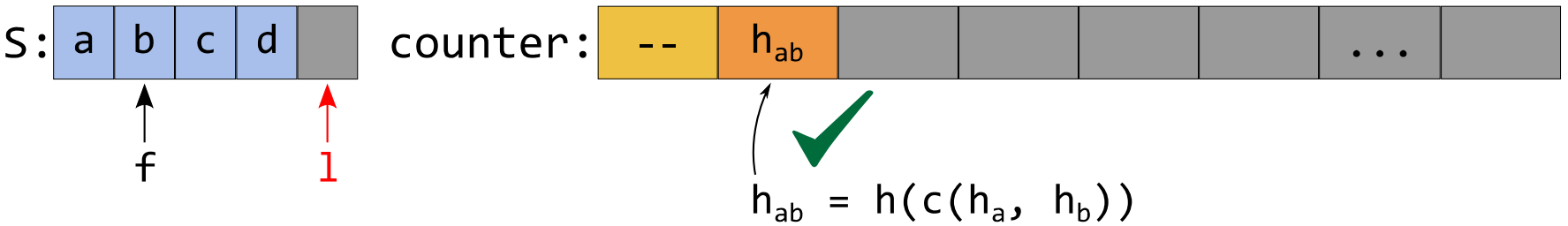

S:
a
b
c
d

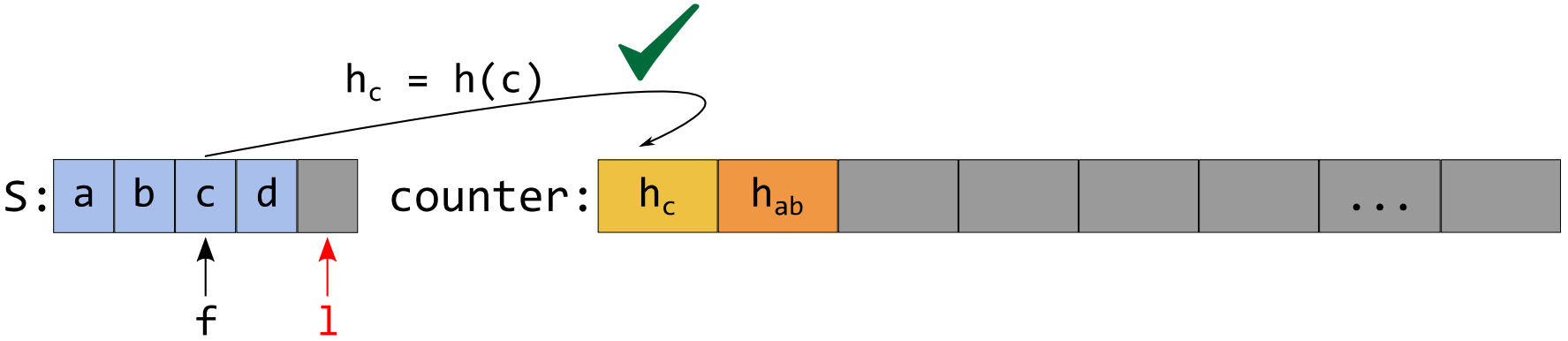

S:
a
b
c
d
counter:
...
f
l

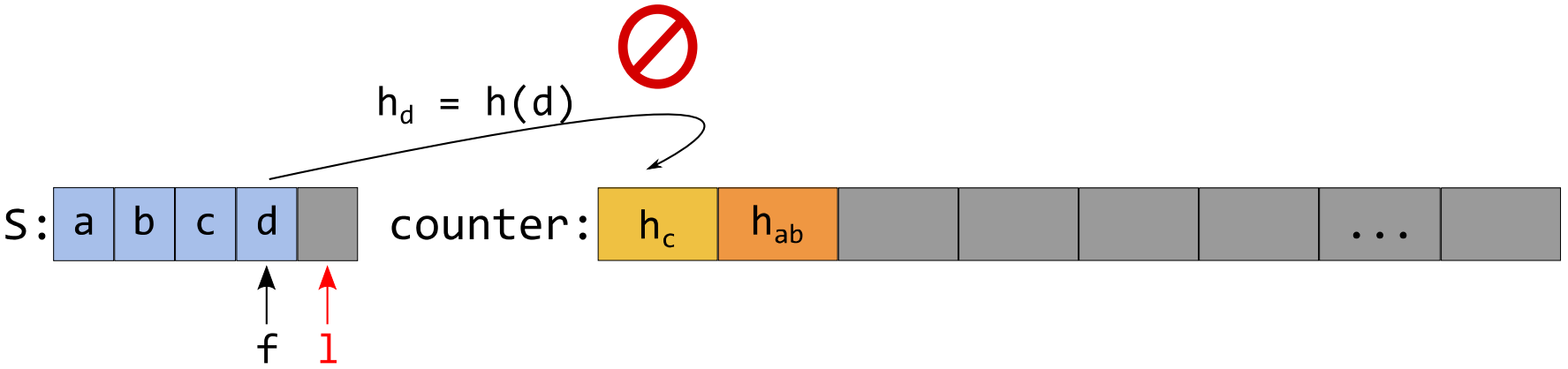

$h_a = h(a)$
S:
a
b
c
d
counter:
h_a
...
f
l

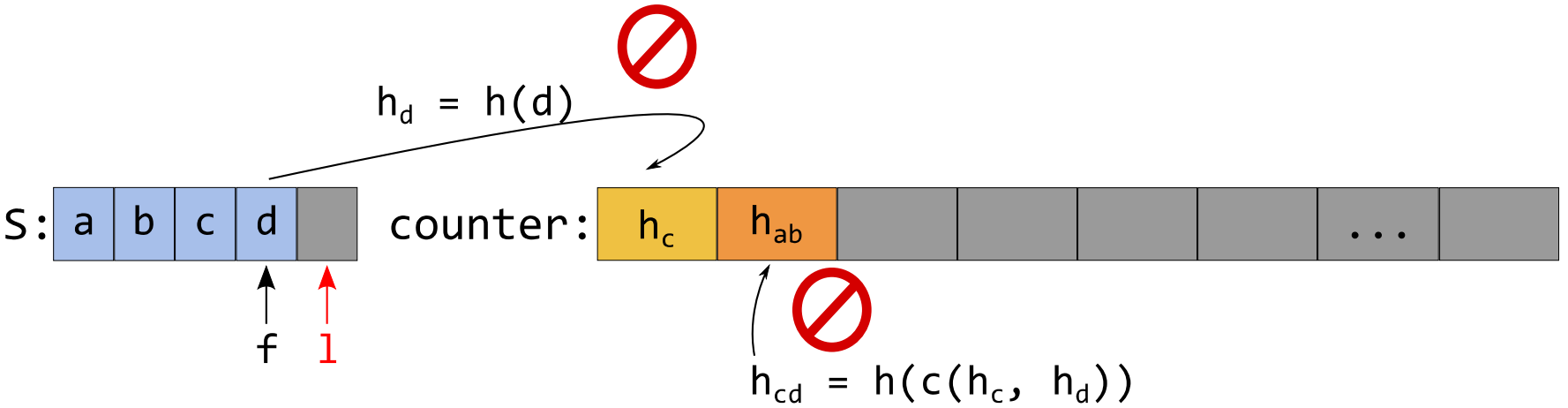

$h_b = h(b)$
S:
a
b
c
d
counter:
h_a
...
f
l

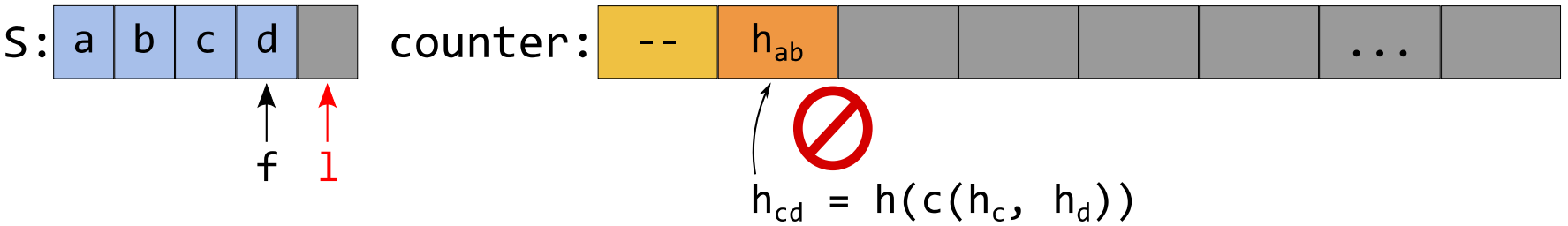

$h_b = h(b)$
S:
a
b
c
d
counter:
h_a
h_{ab}
...
f
l
$h_{ab} = h(c(h_a, h_b))$

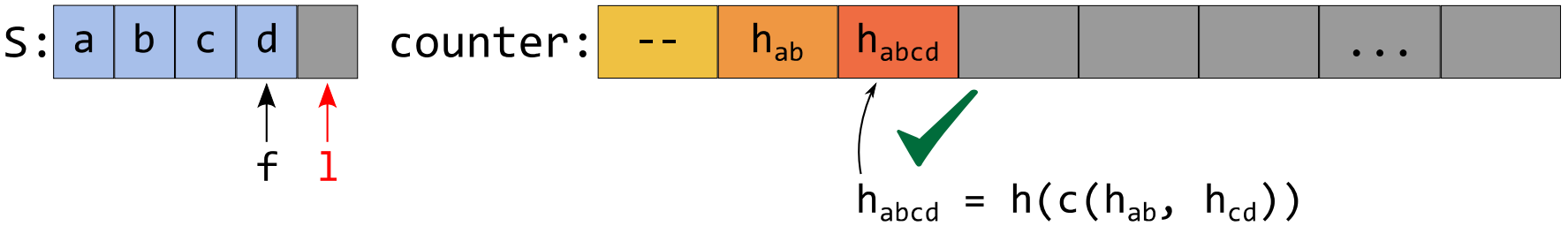

S:
a
b
c
d
f
l
counter:
--
h_{ab}
...
$h_{ab} = h(c(h_a, h_b))$

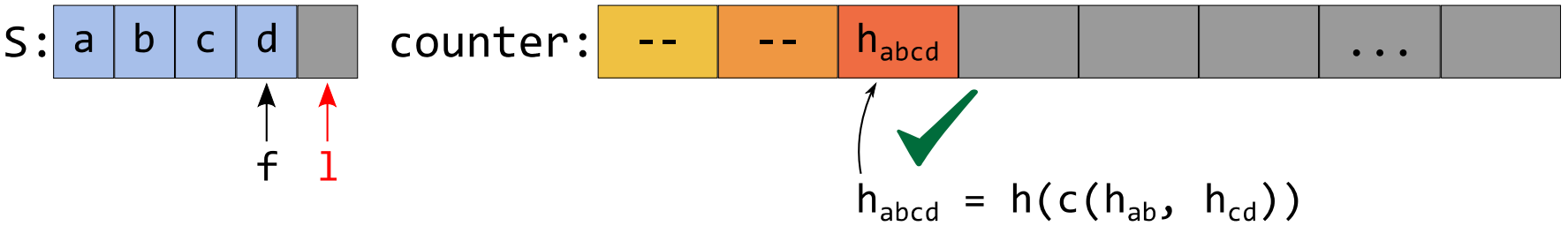

$h_c = h(c)$
S:
a
b
c
d
counter:
h_c
h_{ab}
...
f
l

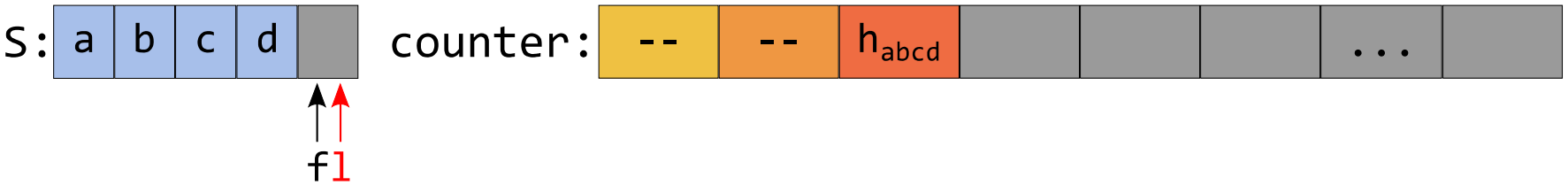

$h_d = h(d)$
S: a b c d
counter: h_c h_{ab} ...
f
l


$h_d = h(d)$
S: a b c d
counter: h_c h_{ab} ...
f
l
$h_{cd} = h(c(h_c, h_d))$

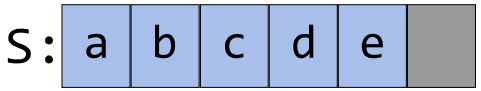

S:
a
b
c
d
f
l
counter:
--
h_{ab}
...
$h_{cd} = h(c(h_c, h_d))$


S:
a
b
c
d
f
l
counter:
--
h_{ab}
h_{abcd}
...
$h_{abcd} = h(c(h_{ab}, h_{cd}))$

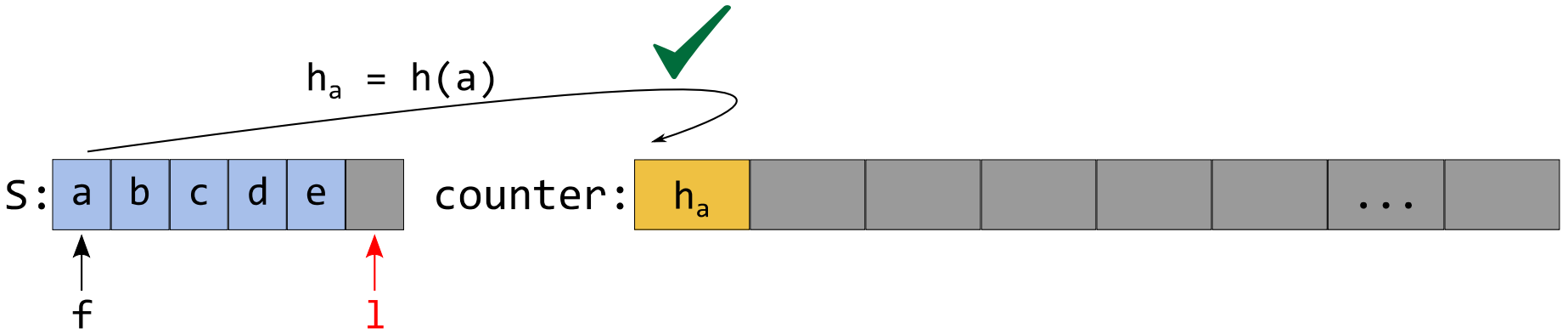

S:
a
b
c
d
f
l
counter:
--
--
h_{abcd}
...
$h_{abcd} = h(c(h_{ab}, h_{cd}))$

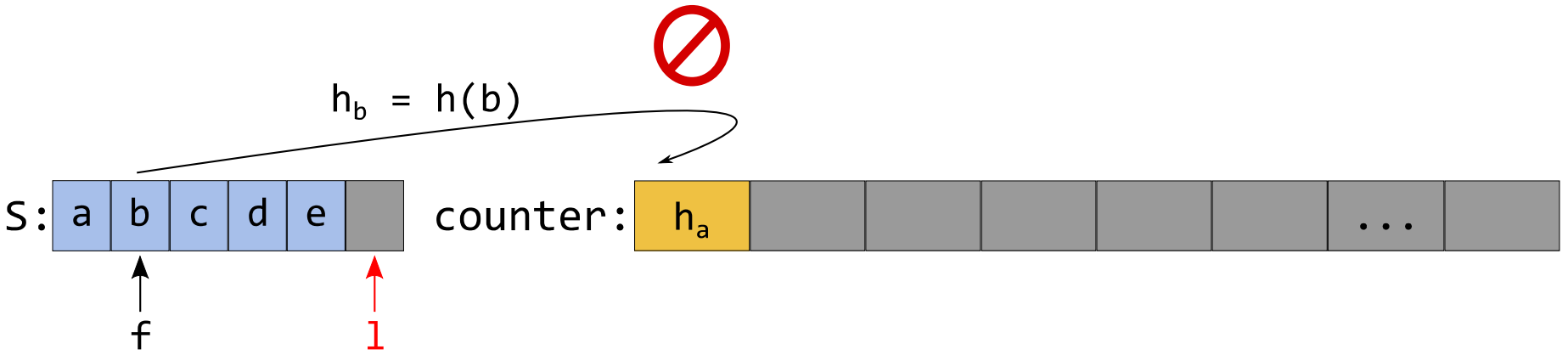

S:
a
b
c
d
counter:
--
--
h_{abcd}
...
f1

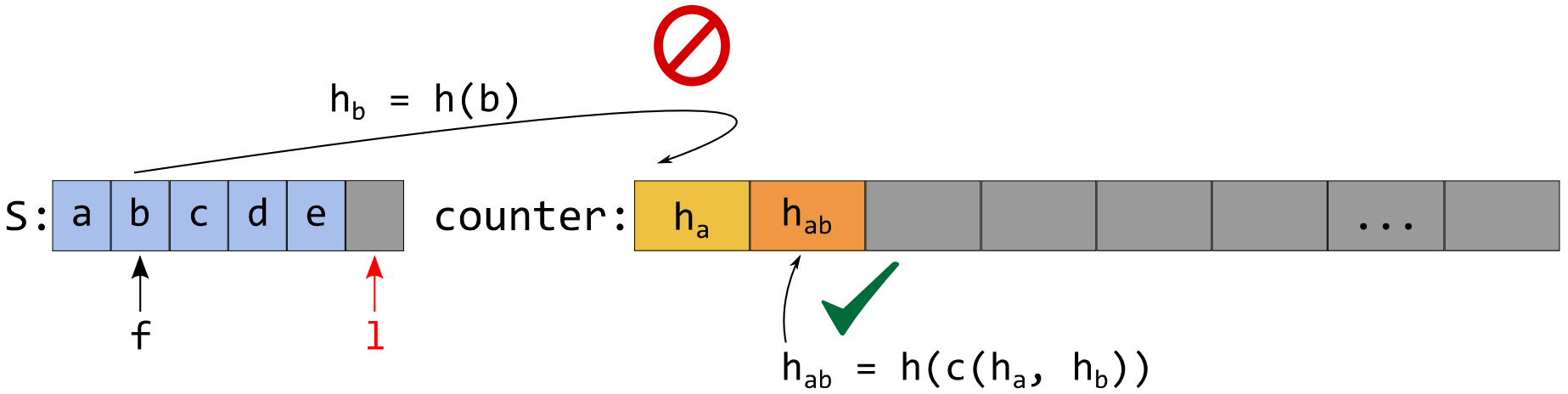

S:
a
b
c
d
counter:
--
--
h_{abcd}
...

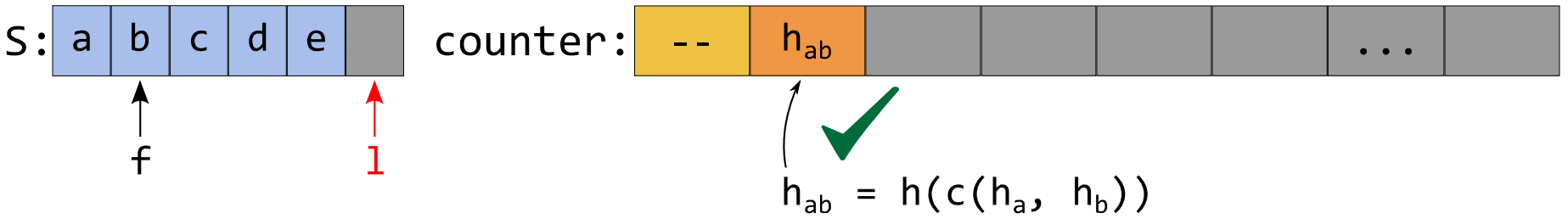

S:
a
b
c
d
e

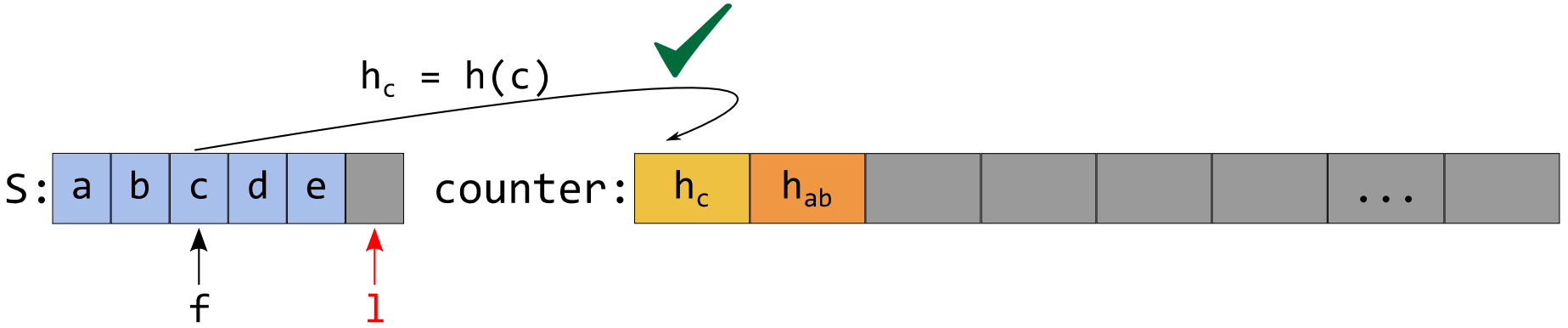

S:
a
b
c
d
e
counter:
...
f
l

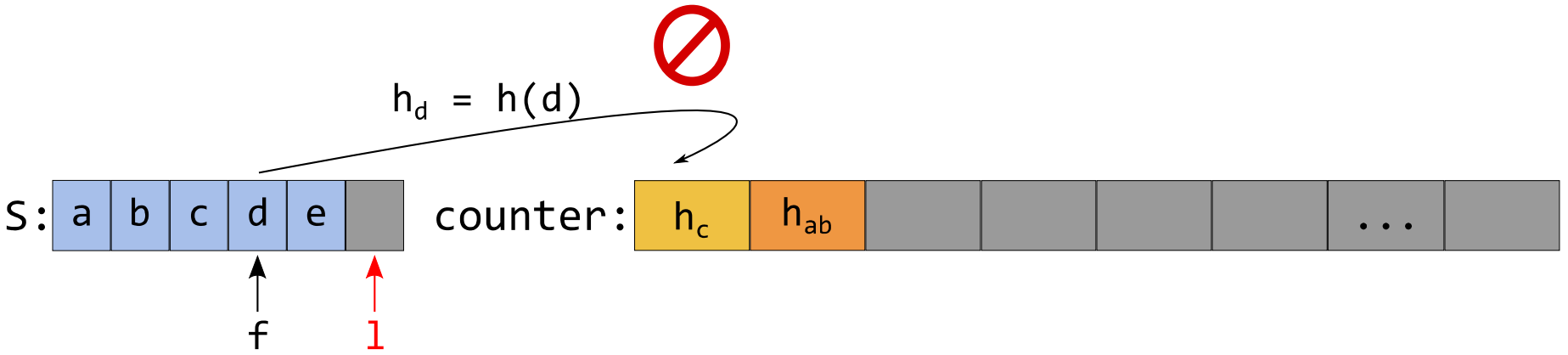

$h_a = h(a)$
S:
a
b
c
d
e
counter:
h_a
...
f
l

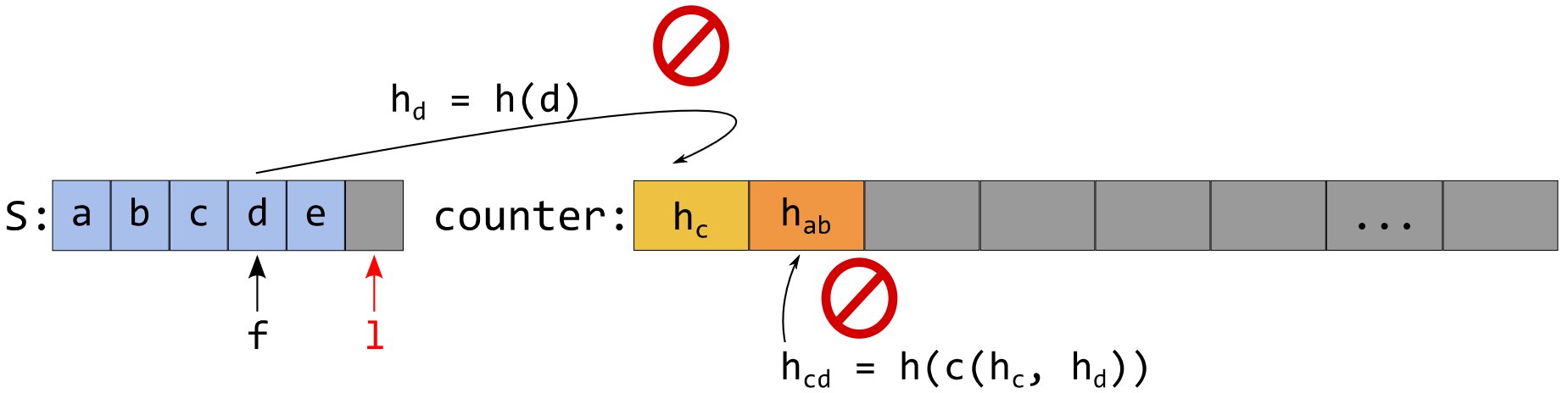

$h_b = h(b)$
S:
a
b
c
d
e
counter:
h_a
...
f
l

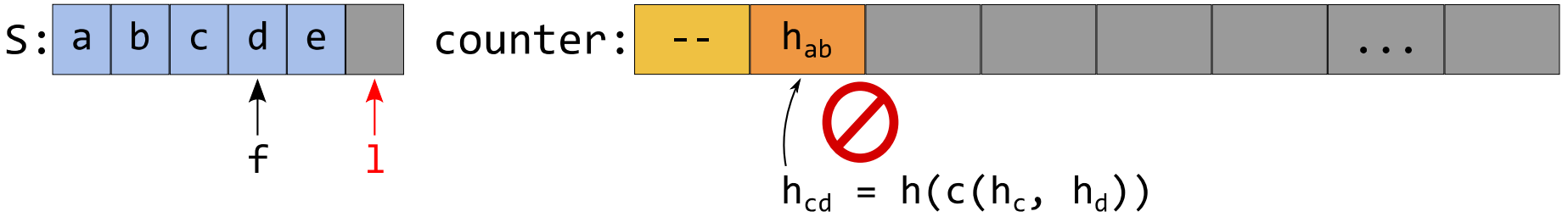

$h_b = h(b)$
S:
a
b
c
d
e
counter:
h_a
h_{ab}
...
f
l
$h_{ab} = h(c(h_a, h_b))$

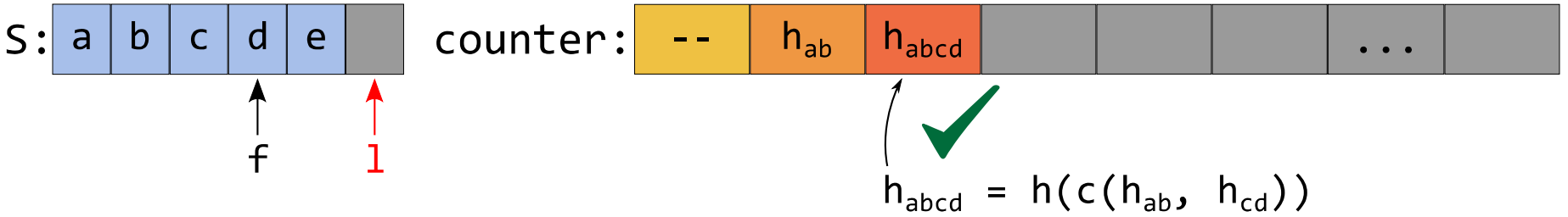

S:
a
b
c
d
e
counter:
--
h_{ab}
...
f
l
$h_{ab} = h(c(h_a, h_b))$

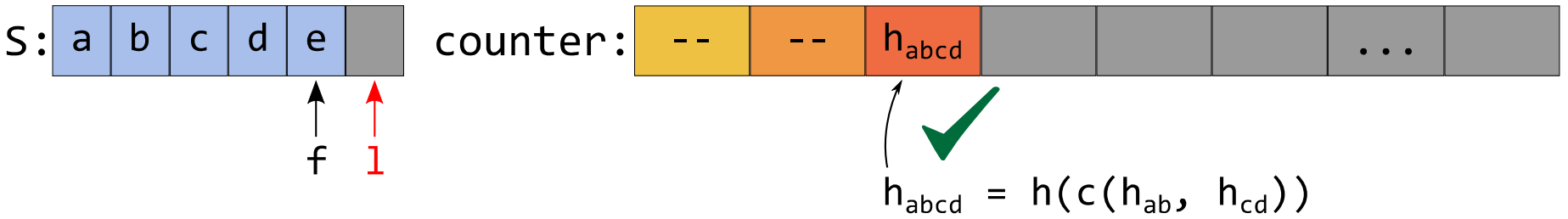

$h_c = h(c)$
S: a b c d e
counter: h_c h_{ab} ...
f
l

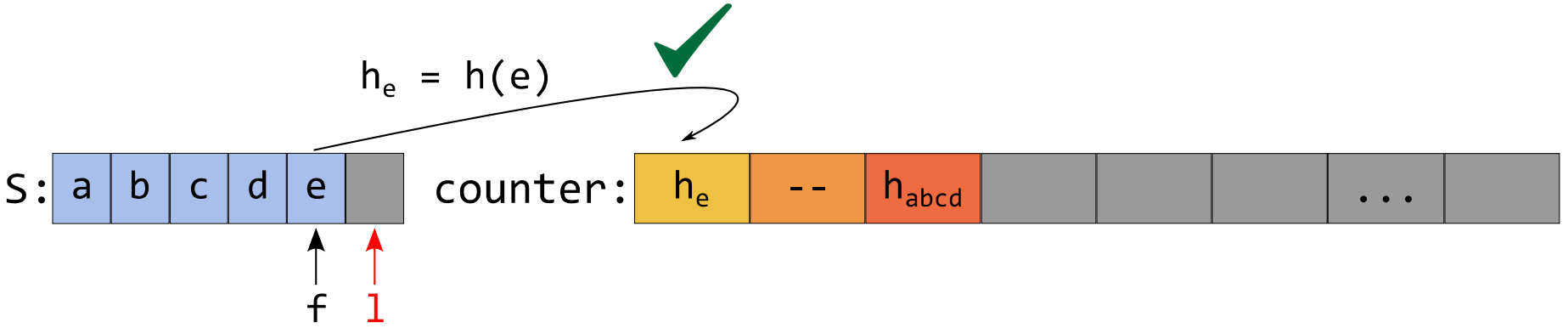

$h_d = h(d)$
S:
a
b
c
d
e
counter:
h_c
h_{ab}
...
f
l

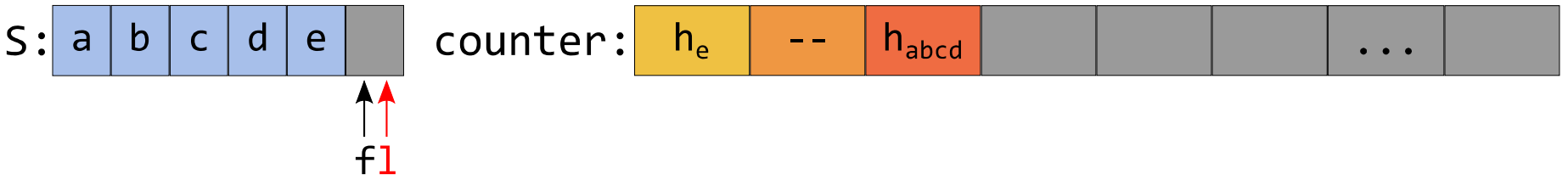

$h_d = h(d)$
S:
a
b
c
d
e
counter:
h_c
h_{ab}
...
f
l
$h_{cd} = h(c(h_c, h_d))$

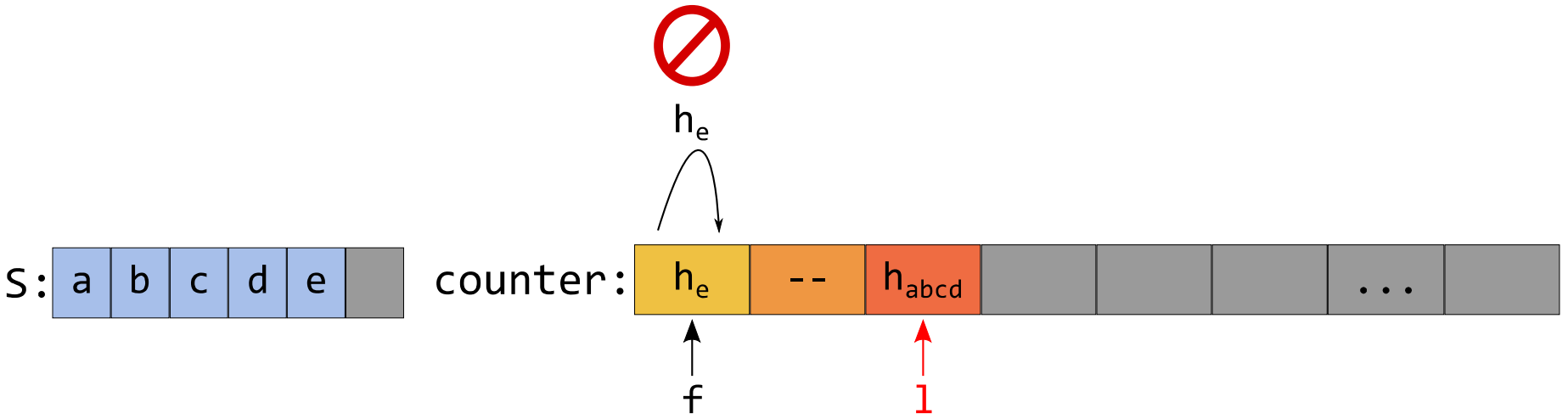

S:
a
b
c
d
e
f
l
counter:
--
h_{ab}
...
$h_{cd} = h(c(h_c, h_d))$

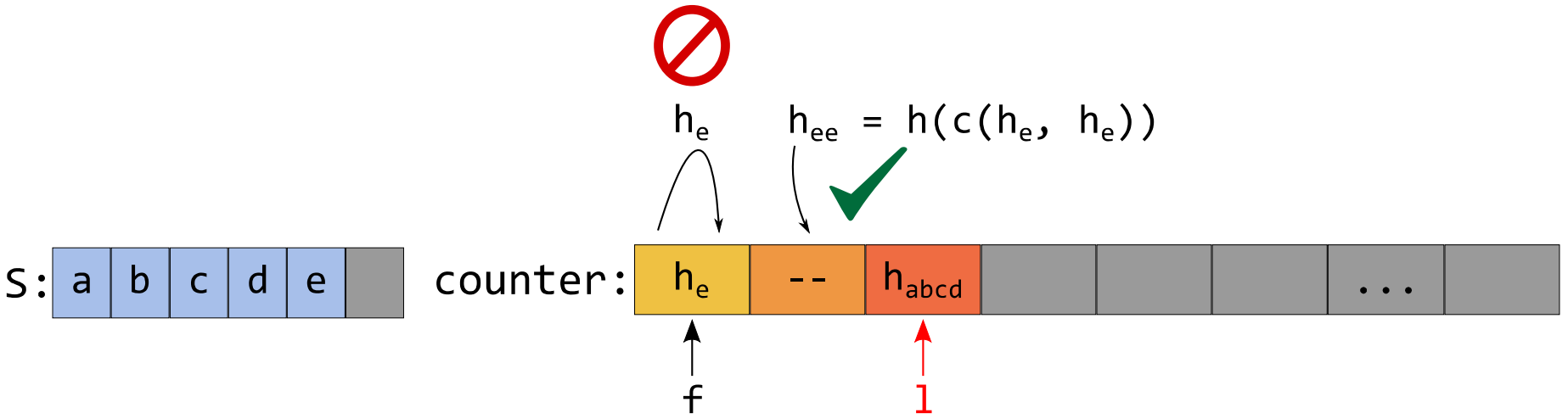

S:
a
b
c
d
e
f
l
counter:
--
h_{ab}
h_{abcd}
...
$h_{abcd} = h(c(h_{ab}, h_{cd}))$

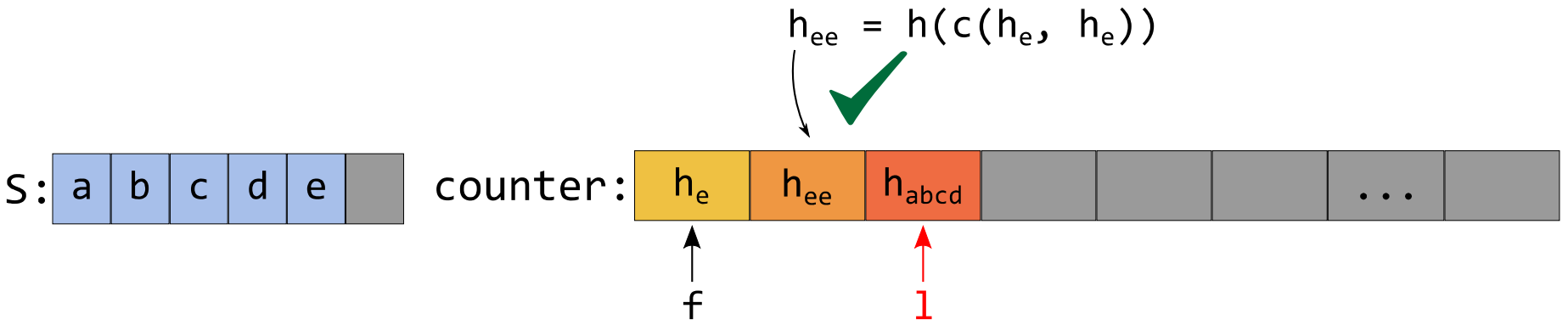

S:
a
b
c
d
e
f
l
counter:
--
--
h_{abcd}
...
$h_{abcd} = h(c(h_{ab}, h_{cd}))$

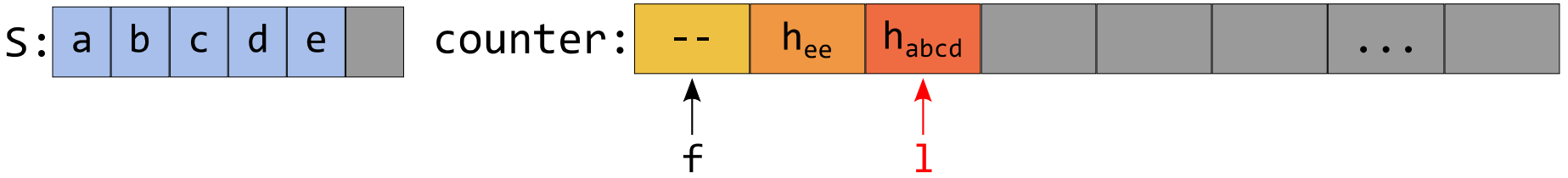

$h_e = h(e)$
S:
a
b
c
d
e
counter:
h_e
--
h_{abcd}
...
f
l

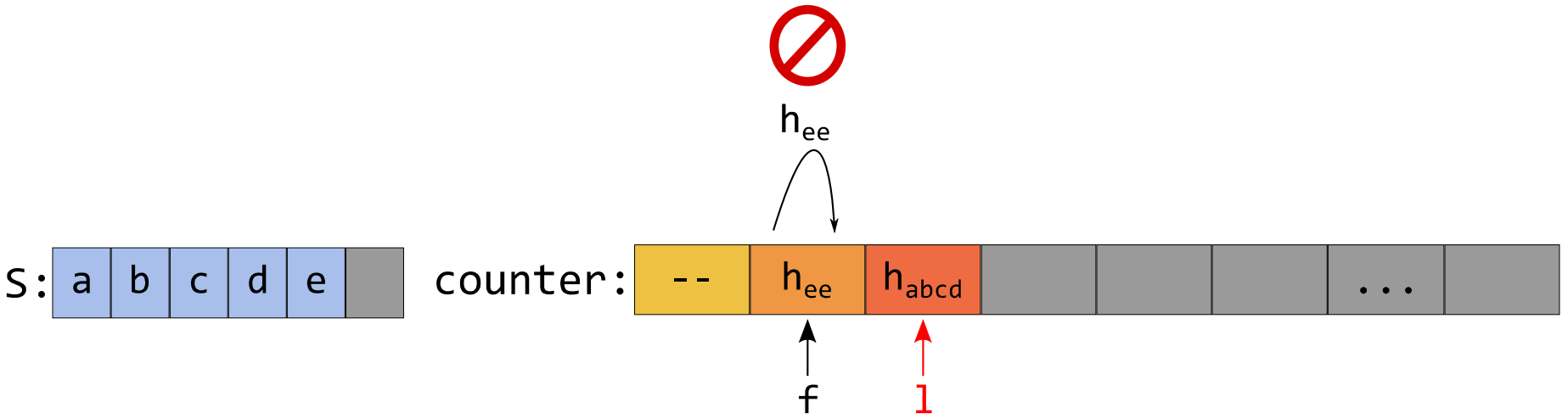

S:
a
b
c
d
e
counter:
h_e
--
h_{abcd}
...
f
1

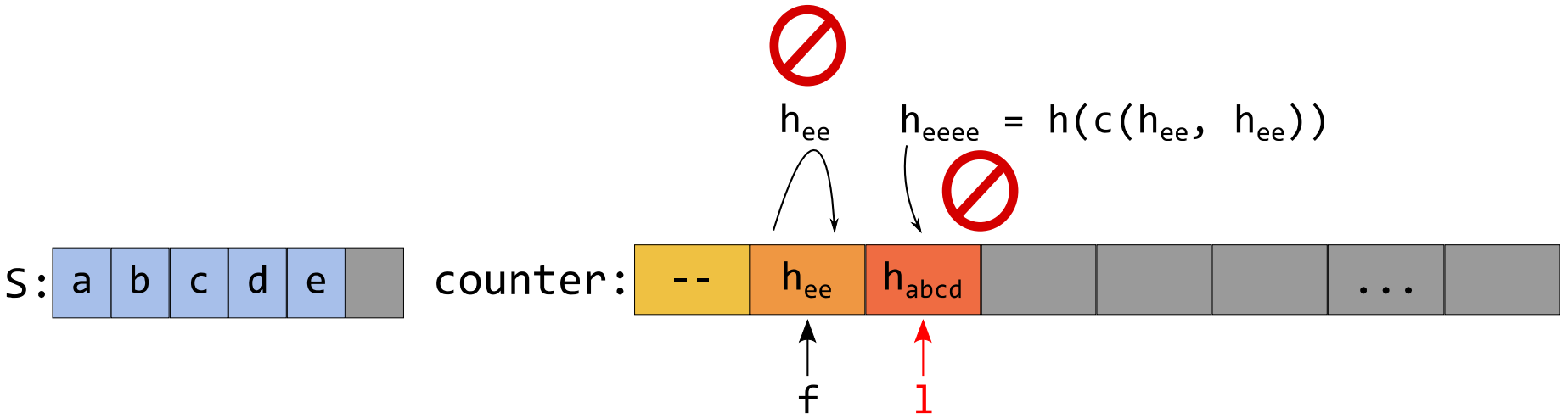

S:
a
b
c
d
e
counter:
h_e
--
h_{abcd}
...
h_e
f
l

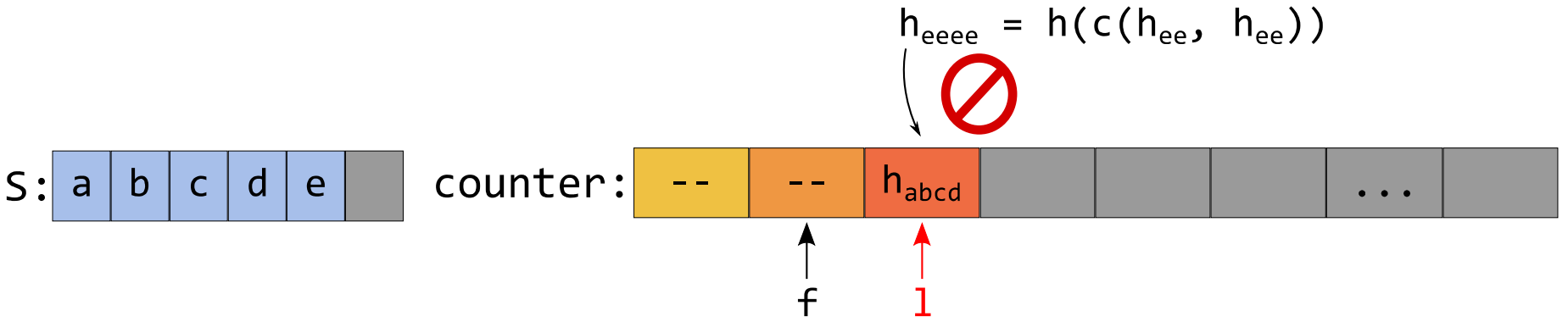

h_e
$h_{ee} = h(c(h_e, h_e))$
S:
a
b
c
d
e
counter:
h_e
--
h_{abcd}
...
f
l

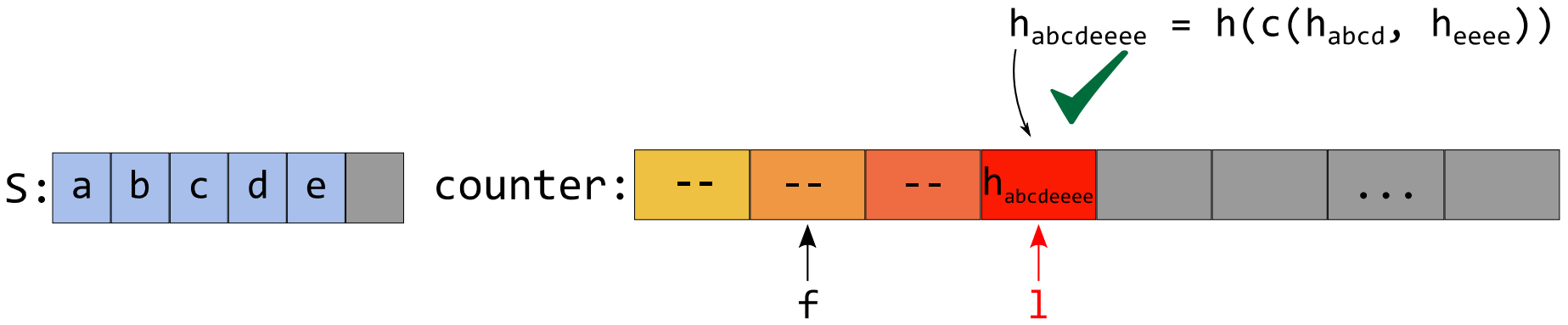

$h_{ee} = h(c(h_e, h_e))$
S:
a
b
c
d
e
counter:
h_e
h_{ee}
h_{abcd}
...
f
l

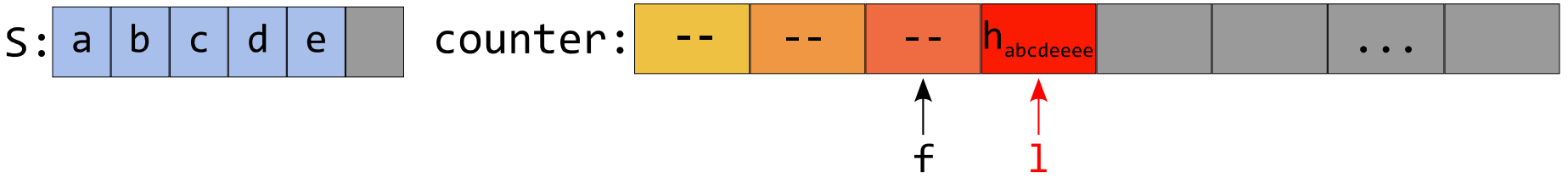

S:
a
b
c
d
e
counter:
--
h_{ee}
h_{abcd}
...
f
l

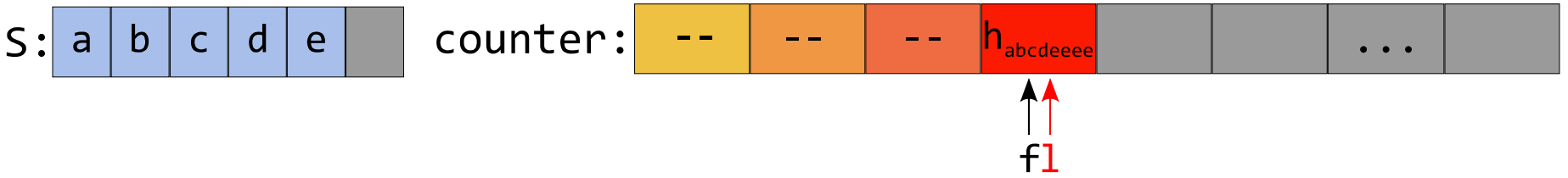

S:
a
b
c
d
e
counter:
--
h_{ee}
h_{abcd}
...
h_{ee}
f
l

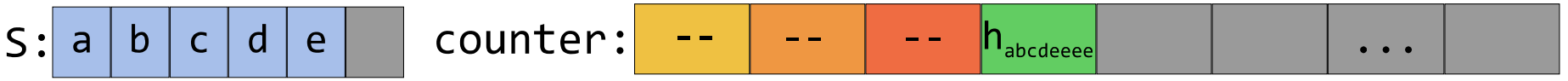

h_{ee}
$h_{eeee} = h(c(h_{ee}, h_{ee}))$
S:
a
b
c
d
e
counter:
--
h_{ee}
h_{abcd}
...
f
l


$h_{eeee} = h(c(h_{ee}, h_{ee}))$
S:
a
b
c
d
e
counter:
--
--
h_{abcd}
...
f
l


$h_{abcdeeee} = h(c(h_{abcd}, h_{eeee}))$
S:
a
b
c
d
e
counter:
--
--
--
$h_{abcdeeee}$
...
f
l


S:
a
b
c
d
e
counter:
--
--
--
h
abcdeeee
...
f
l


S:
a
b
c
d
e
counter:
--
--
--
h
abcdeeee
...
f
l


S:
a
b
c
d
e
counter:
--
--
--
h
abcdeeee
...

La informática (ciencias de la computación) nació de este resurgimiento de la matemática en Europa.

Como programadores, somos ***herederos*** de esta tradición.


Gracias!
Fernando Pelliccioni
@ferpelliccioni
componentsprogramming.com

Bibliografía:

- Antonopoulos, Andreas M. (2010). *Mastering Bitcoin*. O'Reilly.
- Katz, Victor J. (2009). *A History of Mathematics: An Introduction* (3rd ed.). Boston: Addison-Wesley.
- Knuth, Donald E. (2007). *The Art of Computer Programming. Vol. 3: Sorting and Searching*. Boston: Addison–Wesley.
- Stepanov, Alexander, and Daniel Rose. (2015). *From Mathematics to Generic Programming*. Boston: Addison–Wesley Professional.
- Stepanov, Alexander, and Paul McJones. (2009). *Elements of Programming*. Boston: Addison–Wesley Professional.

Versionado:

- V0.1 – 2017-05-10 – versión inicial