According to the decompilation of the Ciso Vigenere hash algorithm, when the password length is less than 16 the idea behind Ciso Vigenere hash algorithm is:
Let p be the password that the user types.
Let hp be the hardcoded password in the code of Packet Tracer.
Let lp be the length of the user input password.
Let h be the hash value obtained from the custom algorithm.
So that:

$\forall h \forall lp \forall hp[(hp =$
$(d, s, f, d, ;, k, f, o, A, ,, ., i, y, e, w, r, k, l, d, J, K, D, H, S, U, B, s, g, v, c, a, 6, 9, 8, 3, 4, n, c, x, v),$
0 <lp <16,
$\text{h}_0 = 0,$
$\text{h}_1 = 8,$
$\text{h} =$

$$\Sigma_{i=2}^{lp} \begin{cases} ((p_i \oplus hp_{8+i}) \ggg 4) + 0x30, & \text{if } (p_i \oplus hp_{i+8} \wedge 0xfffffff0 < 0xa0) \text{ and if } i \equiv 0 \pmod 2 \\ ((p_i \oplus hp_{8+i}) \ggg 4) + 0x37, & \text{if } (p_i \oplus hp_{i+8} \wedge 0xfffffff0 \geq 0xa0) \text{ and if } i \equiv 0 \pmod 2 \\ ((p_i \oplus hp_{8+i}) \wedge 0xf) + 0x30, & \text{if } (p_i \oplus hp_{i+8} \wedge 0xf < 0x0a) \text{ and if } i \equiv 1 \pmod 2 \\ ((p_i \oplus hp_{8+i}) \wedge 0xf) + 0x37, & \text{if } (p_i \oplus hp_{i+8} \wedge 0xf \geq 0x0a) \text{ and if } i \equiv 1 \pmod 2 \end{cases}$$

$) \implies \nexists p[p = \mathbf{rev}(h)]$
(0)

So let's split each sub steps of the algorithm. In this wayt, we could start prooving that if $P \implies Q$ and if $Q \implies R$ then $P \implies R$
So for any P so that:

$$\text{h} = \Sigma_{i=2}^{lp} \begin{cases} (p_i \oplus hp_{i+8} \wedge 0xfffffff0 < 0xa0) \text{ if } i \equiv 0 \pmod 2 \\ (p_i \oplus hp_{i+8} \wedge 0xfffffff0 \geq 0xa0) \text{ if } i \equiv 0 \pmod 2 \\ (p_i \oplus hp_{i+8} \wedge 0xf < 0x0a) \text{ if } i \equiv 1 \pmod 2 \\ (p_i \oplus hp_{i+8} \wedge 0xf \geq 0x0a) \text{ if } i \equiv 1 \pmod 2 \end{cases}$$

$) \implies \nexists p[p = \mathbf{rev}(h)]$
(0)

So for any Q so that:

$$\text{h} = \Sigma_{i=2}^{lp} \begin{cases} (p_i \oplus hp_{i+8} \land 0xfffffff0 < 0xa0), & \text{if } i \equiv 0 \pmod 2 \\ (p_i \oplus hp_{i+8} \land 0xfffffff0 \geq 0xa0) & \text{if } i \equiv 0 \pmod 2 \\ (p_i \oplus hp_{i+8} \land 0xf < 0x0a), & \text{if } i \equiv 1 \pmod 2 \\ (p_i \oplus hp_{i+8} \land 0xf \geq 0x0a), & \text{if } i \equiv 1 \pmod 2 \end{cases}$$

$) \implies \forall p[p = \mathbf{rev}(h)]$
(0)

Let's start by prooving

$\forall h \forall lp \forall hp[(hp =$
$(d, s, f, d, ;, k, f, o, A, ,, ., i, y, e, w, r, k, l, d, J, K, D, H, S, U, B, s, g, v, c, a, 6, 9, 8, 3, 4, n, c, x, v),$
0 <lp <16,
$\text{h}_0 = 0,$
$\text{h}_1 = 8,$

h =

$$\Sigma_{i=2}^{lp} \begin{cases} ((p_i \oplus hp_{8+i}) \ggg 4) + 0x30, & \text{if } (p_i \oplus hp_{i+8} \land 0xfffffff0 < 0xa0) \text{ and if } i \equiv 0 \pmod 2 \\ ((p_i \oplus hp_{8+i}) \ggg 4) + 0x37, & \text{if } (p_i \oplus hp_{i+8} \land 0xfffffff0 \geq 0xa0) \text{ and if } i \equiv 0 \pmod 2 \\ ((p_i \oplus hp_{8+i}) \land 0xf) + 0x30, & \text{if } (p_i \oplus hp_{i+8} \land 0xf < 0x0a) \text{ and if } i \equiv 1 \pmod 2 \\ ((p_i \oplus hp_{8+i}) \land 0xf) + 0x37, & \text{if } (p_i \oplus hp_{i+8} \land 0xf \geq 0x0a) \text{ and if } i \equiv 1 \pmod 2 \end{cases}$$

$) \implies \nexists p[p = \mathbf{rev}(h)]$
(0)

I/ substraction to reverse the addition
$\forall x[(x = y + z) \implies (y = ez)]$ then it follow that as the previous part of the function contains: $h = x + 0x30$, then $h - 0x30 = x$ so
$\exists rev(h)[rev(H(p)) = p - 0x30]$
II/ exclusive or
According to the Karnaught table at: https://fr.wikipedia.org/wiki/Table$_{d}e_{v}$

$$Thenas$$

xlat $\oplus xlat = 0$, and as $p \oplus 0 = p$, we know that the original password $p = xlat \oplus h$.

III/ rotating 4 first to 4 last bits
$\forall x[(x \ggg y) \implies (x \lll y = x)]$.

$$Thenas$$

z = (x $\ggg y) = (x \lll y)$, we know that the original password $p = H(p) \lll 4$.

IV/ unmasking different signatures (recurrent marks) in the hash
In the previous chapter one 'I/ substraction to reverse the addition', we told we can reverse the previous addition. We still need to guess which addition/substraction has been done previously.

As both addition values are made depending of:
“‘ if (password[left]  0xfffffff0 ¡ 0xa0) -¿ password[left] + 0x30 or else password[left] + 0x37 if (password[right]  0x0000000f ¡ 0x0a) -¿ password[right] 0x0f + 0x30 or else password[right]  0x0f + 0x37 “‘
Then we have proven that:
$hp =$
$(d, s, f, d, ;, k, f, o, A, ,, ., i, y, e, w, r, k, l, d, J, K, D, H, S, U, B, s, g, v, c, a, 6, 9, 8, 3, 4, n, c, x, v) \implies$
$(\forall x \in hp[0 \geq x0 \geq 256 \implies x \in hp])$
then:

Let p be the password that the user types.
Let hp be the hardcoded password in the code of Packet Tracer.
Let lp be the length of the user input password.
Let h be the hash value obtained from the custom algorithm.
So that:

$\forall h \forall lp \forall hp[(hp \in N \wedge 0 \geq hp,$
0 <lp <16,
$\mathrm{h}_0 = 0,$
$\mathrm{h}_1 = 8,$
h =

$$\Sigma_{i=2}^{lp} \begin{cases} (((p_i \oplus hp_{i+8}) \lll 4) - 0x30), & \text{if } p_i < 0xa0 \text{ and if } i \equiv 0 \pmod 2 \\ (((p_i \oplus hp_{i+8}) \lll 4) - 0x37), & \text{if } p_i \geq 0x0a0 \text{ and if } i \equiv 0 \pmod 2 \\ (((p_i \oplus hp_{i+8}) \wedge 0xfffffffff0) - 0x30), & \text{if } p_i < 0x0a \text{ and if } i \equiv 1 \pmod 2 \\ (((p_i \oplus hp_{i+8}) \wedge 0xfffffffff0) - 0x37), & \text{if } p_i \geq 0x0a \text{ and if } i \equiv 1 \pmod 2 \end{cases}$$

$) \implies \forall p[p = \mathbf{rev}(h)]$