To prove that LTFS is NP-Complete, it needs to satisfy two conditions:

1) Show that LTFS is in NP
2) Show that there exists a mapping reduction from an NP-Complete problem to LTFS

Claim 1: LTFS is in NP

Proof: Assume we have a subset S of Graph G with $size \geq k$. We can iterate each triple u, v, w in this subset S. And for each triple, we can very easily check if this particular triple has at least one absent edge. Then two possibility exist for each triple that we iterate:

a) There EXISTS an absent edge among the three vertices → continue iteration
b) There DOESN'T EXIST an absent edge → the subset S is not a LTFS

Explanation: The verifier for determining whether a subset S is LTFS is an exhaustive search, and it will guarantee to produce a result. And the algorithm will run in polynomial time, therefore LTFS is in NP.
(COMPLETE)

Claim 2: There is a mapping reduction

---

1) We can reduce Independent Set (IS) to LTFS. First, we need to build a mapping from an instance of $IS\ (G, k)$ to an instance of $LTFS\ (G_i, k_i)$. G has an independent set of $size \geq k$ if and only if G_i has a triangle-free subset of $size \geq k_i$.
2) Assume we have an instance (G, k) of IS where $G = (V, E)$ $where\ V = n\ and\ E = m.$ The basic idea is that we will replace every edge in G with a triangle. Do the following:
    a) Create new vertex v_{new} for every edge $e = (u, v) \in E$. Store all these new vertices to a set V_{new}
    b) For every new vertex v_{new}, create two new edges $e_{new1}\ and\ e_{new2}$, where $e_{new1} = (u, v_{new})$ and $e_{new2} = (v, v_{new})$. Store all the new edges to a set E_{new}
    c) Copy the content of G to G_i, and add all the new vertex set V_{new} and edge set E_{new} to G_i, and now $G_i = (V \cup V_{new}, E \cup E_{new})$
    d) We need to update k_i as well. Set $k_i = m + k$. Now G_i is a triangle-free subset of $size \geq m + k$

We need to prove these two statements:

1) If G has an independent set with $size \geq k$, then G_i has a triangle-free subset with $size \geq m + k$.
2) If G_i has a triangle-free subset with $size \geq m + k$, then G has an independent set with $size \geq k$.

Proof of statement (1):

---

claim (a): S_{IS} is an independent set of G with $size \geq k \Rightarrow (1)$
Assume that there exists an independent set $S_{IS} \in G$ with $size \geq k$.

Claim (b): S_{IS} is an independent set of G_i and S_{IS} is triangle-free in $G_i \Rightarrow (2)$

Proof: Because S_{IS} is an independent set of G, and when we are building the G_i graph, we didn't add any edge between vertices in G. Then S_{IS} should be an independent set in G_i, in graph G_i, for any two vertices in S_{IS}, there will be no edge connecting the two.

Claim (c) : S_i is triangle-free subset in G_i with $size \geq m + k \Rightarrow (3)$

Proof: Since we know that V_{new} is the set that contains the new vertex for every edge in G, therefore, $|V_{new}| = m$. There should exist a $S_i = S_{IS} \cup V_{new}$, and because S_{IS} and V_{new} are disjoint sets, that is to say they are not connected by any edge, so $|S_i| \geq k + m$. S_i should also be triangle-free in G_i. The reason is as the following:

→ Because V_{new} has no edge between them, S_{IS} is triangle-free in G_i, and the fact that S_i is composed of the union of S_{IS} and V_{new}, S_i has 2 vertices from S_{IS} and 1 vertex from V_{new}. The addition of any vertex in V_{new} will not create a triangle. We know that for each vertex V_{new} with $e = (u, v)$, because S_{IS} is an *independent set* of G_i, u and v should not be both present in S_{IS}. If both u and v are not present in S_{IS}, then adding V_{new} to S_i will not create a triangle because that particular vertex is not connected to any vertices in S_{IS}. If one of u and v is present in S_{IS}, it will not create a triangle either because in order to create a triangle both u and v needs to be present in S_{IS}. Therefore S_i is also triangle-free in G_i with $size \geq m + k$!

Proof of statement (2):

Claim (a): G_i has a triangle-free subset S_i with $size \geq m + k \Rightarrow (4)$

Assume there exists some triangle-free subset S_i of graph G_i, with $size \geq m + k$, and we choose any triangle-free subset S_i from G_i with $size \geq m + k$.

Claim (b) : If $S_i = S_{some\ set} \cup V_{new}$ (S_i contains the whole set V_{new}), then that $S_{some\ set}$ is an independent set of G with $size \geq k \Rightarrow (5)$

Proof: If S_i contains the whole set V_{new}, then $S_{some\ set}$ should be an independent set of G_i. If $S_{some\ set}$ is not IS of G_i, there will be two vertices u and v that share an edge, then set S_i would end up with $\{u, v, V_{new}\}$ with their edges connected to each other, since previously we have shown that we are adding $e_{new1} = (u, v_{new})$ and $e_{new2} = (v, v_{new})$ to form G_i, if somehow vertex u and v are connected, the triple would form a triangle, which contradicts to our statement that S_i is a triangle-free subset of G_i. Therefore, $S_{some\ set}$ is an independent set of G with $size \geq |S_i| - |V_{new}| = k$.

Claim (c) : if $S_i \cap V_{some} \neq V_{new}$ (S_i doesn't contains the entire whole set V_{new}), then we will always find a set that $S_{another} = S_{some\ set} \cup V_{new}$, and then that $S_{some\ set}$ is an independent set of G with $size \geq k$

Proof: If S_i doesn't contains the entire whole set V_{new}, we can always find another triangle-free subset $S_{another}$ that contains the entire whole set of V_{new} with $S_{another} = S_{some\ set} \cup V_{new}$. For every vertex $v_e \in V_{new}$ but $\notin S_i$ with $e = (u, v)$:

1) If at least one of the vertex u and v is absent from S_i → add v_e to S_i
2) If both u and v are present in S_i, → remove u from S_i and add v_e to S_i

Using this method, we can always find add new vertices V_{new} into the S_i without creating a triangle, and when we get such a $S_{another}$ and add all the V_{new}, we can just use the rationale from claim (b) to show that $S_{some\ set}$ is an independent set of G with $size \geq k$

Conclusion: We made an reduction from an NP-Complete problem to LTFS problem in polynomial time, therefore, the LTFS problem is NP-Complete (COMPLETE)