agents between the stranger Agt_{k-m} and the original requester, Agt_{k-m} can hardly infer the identity of the original requester based on information he learns. When $m = k$, all ACs must be exchanged between friends only. When $m = 1$, the last agent is the only one who is in social tie of the original requester, which is the case in our example.

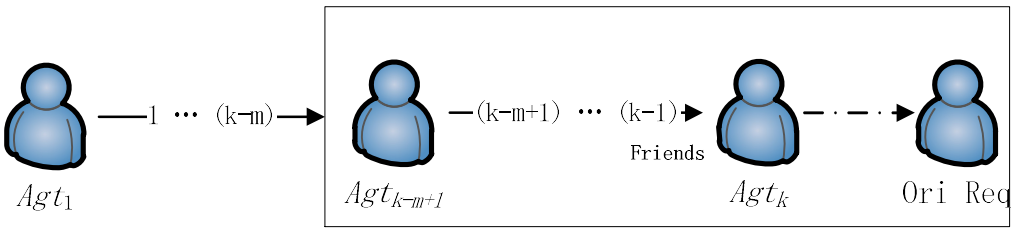

Figure 4.4: Friends-Obfuscation Distance

However, friends encounter each other rarely, so having a large *m* increases the difficulty of distributing ACs. In this work, we assign m to 1.

4.5.3. Distributing Segment

We again use the example in Figure 4.1. U_a generates 3 ACs (i.e. AC_a^1, AC_a^2 and AC_a^3). It exchanges these ACs with U_b, then U_b exchanges them with U_c and so on. At last, all of them reach the original requester U_d. In this case, U_d has several ACs whose paths (the list of agents) are the same, so U_d has no choice but to choose one. In other words, since using these ACs results in the same reply path, U_d is can simply choose an AC based on specific requirement it might have (e.g. the number of possible encounters of an agent).

If agents exchange ACs with different users, the above problem will not happen. We use the system parameter distributing segment, Seg, to avoid giving all ACs to the same user. Each user maintains Seg number of *Distributing AC List*s (*DL*s). It puts received *distributing* ACs in one of those *DL*s randomly. If Agt_i exchanges ACs with another user, Agt_i selects one of his *DL*s, and only ACs in that *DL* will be exchanged with that user.

For example, we assign $Seg = 2$ so that each user has two *DL*s (i.e., *DL*1 and *DL*2).