1. Implement authenticated perfect links using digital signature primitive

   使用数字签名原语的认证完美点对点链路

```
Implements: AuthPerfectPointToPointLinks, instance al.
Uses: StubbornPointToPointLinks, instance sl.

upon event < al, Init > do
    delivered := ∅;

upon event < al, Send | q, m > do
    σ := sign(self, m);
    trigger < sl, Send | q, [m, σ] >;

upon event < sl, Deliver | p, [m, σ] > do
    if verifysig(p, m, σ) ∧ m ∉ delivered then
        delivered := delivered ∪ {m};
        trigger < al, Deliver | p, m >;
```

2. What happens if the assumption for Δ is not held? Which properties of leader election module are violated? Give an example.

   当Δ（通信、处理时间或时钟漂移的上限）的假设不成立时，可能会违反领导者选举模块（Leader Election Module）的某些属性。以下是具体的分析和一个例子：

   Δ的假设不成立时的影响

   1. **如果Δ不成立**，意味着系统不能保证消息在预定的时间内被传递，处理时间也可能超出预期，时钟可能有不可预测的漂移。这将导致领导者选举算法无法在预期的时间内正确地判断领导者是否崩溃，从而影响其准确性和最终检测。
   2. **准确性（Accuracy）被违反**：
      - **属性LE2（Accuracy）**规定，如果一个进程被选为领导者，那么之前的所有领导者必须已经崩溃。如果Δ不成立，一个领导者可能被错误地认为已经崩溃，导致选举出新的领导者，而实际上前一个领导者可能仍然是正确的。

   假设在一个分布式系统中有三个进程A、B和C。进程A是当前的领导者。

   - 假设由于网络延迟（违反Δ），进程B和C在较长时间内没有收到来自A的心跳消息。
   - B和C可能会错误地判断A已经崩溃，然后启动新的领导者选举过程。
   - 在这个过程中，B可能被选为新的领导者。
   - 然而，A实际上并未崩溃，只是由于网络问题暂时无法通信。这就违反了准确性属性，因为现在有两个没有崩溃的领导者A和B同时存在。

最终检测（Eventual Detection）的不确定性

- **属性LE1（Eventual Detection）**规定，要么没有正确的进程，要么某个正确的进程最终会被选为领导者。
- 如果Δ的假设不成立，最终检测可能无法保证，因为系统无法准确地判断领导者是否崩溃。这可能导致系统无法在合理的时间内选出新的领导者，或者错误地频繁更换领导者。

总结来说，Δ的假设不成立可能导致领导者选举算法的准确性和最终检测能力受到影响，从而影响整个分布式系统的稳定性和可靠性。

3. If we would prefer p2 to be the leader (though it may crash), give an eventual leader detection algorithm that additionally ensuresIf p2 does not crash, then eventually p2 is elected as the leader by every correct process.

   如果我们更喜欢p2作为领导者（尽管它可能崩溃），则给出一个最终的领导者检测算法，该算法还保证如果p2没有崩溃，那么最终p2通过每个正确的过程被选为领导者。

```
# 实现：最终领导者检测器，实例名为Ω。
# 使用：最终完美故障检测器，实例名为◇P；固执点对点链路，实例名为sl。

# 当Ω实例初始化时执行
upon event < Ω, Init > do
    suspected := ∅;  # 初始化一个空的怀疑集合
    leader := ⊥;       # 初始化领导者为未定义

# 当◇P实例怀疑某个进程p崩溃时执行
upon event < ◇P, Suspect | p > do
    suspected := suspected ∪ {p};  # 将进程p加入怀疑集合

# 当◇P实例恢复对某个进程p的信任时执行
upon event < ◇P, Restore | p > do
    suspected := suspected \ {p};  # 将进程p从怀疑集合中移除

# 当p2不在怀疑集合中且当前领导者不是p2，或者p2在怀疑集合中且当前领导者不是未怀疑进程中的最高排名进程时执行
upon (p2 ∈ Π \ suspected and leader ≠ p2) or (p2 ∈ suspected and leader ≠ maxrank(Π \ suspected)) do
    if p2 ∈ Π \ suspected then
        leader := p2;  # 如果p2未被怀疑，则选择p2作为领导者
    else
        leader := maxrank(Π \ suspected);  # 否则，选择未怀疑进程中的最高排名进程作为领导者
    trigger < Ω, Trust | leader >;  # 触发信任事件，通知新的领导者
```