```
def checkBinding(x, invertedVote, trunSize):
    for i in range(2**16):
        i = intToByte(i)
        newX = commitment(invertedVote, i, trunSize)
        if newX == x :
            return True
    return False

def binding_property(size, trunSize):
    san = 0
    for j in range(size):
        myX = commitment(b'\0',generateK(), trunSize)
        if checkBinding(myX, b'\1', trunSize):
            san += 1
    return (san/size) * 100

def checkConcealing(x, vote, trunSize):
    summa = 0
    for i in range(2**16):
        i = intToByte(i)
        newX = commitment(vote, i, trunSize)
        if newX == x :
            summa += 1
    return summa


def FindOutVote(x, turnSize):
    nbrOfOne = checkConcealing(x, b'\1', turnSize)
    nbrOfZero = checkConcealing(x, b'\0', turnSize)
    if nbrOfOne > nbrOfZero :
        return 1
    else:
        return 0

def conceling_property(size, turnSize, MyVote):
    nbrOfVin = 0
    for i in range(size):
        k = generateK()
        commit = commitment(MyVote, k, turnSize)
        what = FindOutVote(commit, turnSize)
        if what == 1:
            opponentVote = b'\0'
        else:
            opponentVote = b'\1'
        if MyVote != opponentVote:
            nbrOfVin += 1
    return (nbrOfVin/size) *100
```