1 ResultVerification Smart Contract

Algorithm 1: ResultVerification Smart Contract

```
Data: Contract Variables:
tags: Mapping(bytes32, Tag)
Data: Struct:
struct Tag
uint A
uint[] wit
Input: Function Parameters:
uint N, uint x, uint[] wit, uint A
Output: Function Return:
uint A_prime
Function accVerify(N, x, wit, A):;
A_prime := (wit[x]**x) % N;
Require: A == A_prime, "Verification failed";
Return: A_prime
Function resultVerify(N, x, resultIds, wit, A):;
Require: msg.sender == address(this), "Access denied";
for i in range(resultIds.length) do
    id := resultIds[i];
    A_prime := accVerify(N, x, wit, A[i]);
    tags[id] := Tag(A_prime, wit);
end
Return: true
```

2 ResultReceiving Smart Contract

Algorithm 2: ResultReceiving Smart Contract

```
   Data: Contract Variables:
   RVC: Address
   DA: Address
   IDA: Address
   resultStorage: Mapping(bytes32, uint[])
   Input: Function Parameters:
   bytes32 token, uint[] resultIds
1  Constructor(rvc, da, ida):;
2  RVC := rvc;
3  DA := da;
4  IDA := ida
5  Function resultRecord(token, resultIds):;
6  Require: msg.sender == RVC, "Access denied";
7  resultStorage[token] := resultIds
8  Function resultGet(token):;
9  Require: msg.sender == DA, "Access denied";
10 Return: resultStorage[token]
```

Variable	**Description**
tags	Mapping storing tags associated with resultIds
Tag	Struct containing A (accumulation value) and wit (witness array)
N	Parameter representing a value
x	Parameter representing an index
wit	Parameter representing an array of witness values
A	Parameter representing an accumulation value
resultIds	Parameter representing an array of result identifiers
RVC	Address variable representing the Result Verification Contract
DA	Address variable representing the Data Analyst
IDA	Address variable representing the Device Administrator
resultStorage	Mapping storing result data associated with tokens
token	Parameter representing a unique token

Table 1: Variable Naming and Descriptions