Bug Report to Peanut Protocol

Findings 1-3 are persistent in both V3 and V4.
Finding 4 is only present in V3.

All code references below are for V4 unless specified otherwise.

Finding #1-3

Lines 235, 281, and 333 are incorrect and will result in an EVM error due to the lengths of `_data` and `_pubKey20Bytes` being restricted to 20 bytes. Solidity's `abi.decode()` function expects a 32-byte encoded data set.

Therefore, all direct external transfers of ERC721 and ERC1155 tokens will fail for failing to implement their expected receiver.

```
    pubKey20: abi.decode(_data, (address)),
    ...
    pubKey20: abi.decode(_data, (address)),
   ...
    pubKey20: abi.decode(_pubKey20Bytes, (address)),
```

Run the following commands to reproduce the bug:

```
$ FOUNDRY_PROFILE=peanut_v3 forge test
```

```
$ FOUNDRY_PROFILE=peanut_v4 forge test
```

Finding #4

Resolved in V4

No impact due to the above finding for ERC1155 batch transfers.

Within V3, should the ERC1155 batch transfer issue be resolved in the previous finding, a batch transfer into this contract would forever lock up that asset within the contract. This is due to the fact that the `onERC1155BatchReceived()` function deposits the token(s) with a `_contractType` of `4`, and the `withdrawDeposit()` function would bypass all transfers and delete the deposit record.

Remedial Measures

To keep the same behavior use solidity's `abi.encode()` to maintain the same process flow and logic with minimal changes. This will pad the 20-bytes data to 32-bytes and not revert.

```
     pubKey20: abi.decode(abi.encode(_data), (address)),
     ...
     pubKey20: abi.decode(abi.encode(_data), (address)),
    ...
     pubKey20: abi.decode(abi.encode(_pubKey20Bytes), (address)),
```

Alternatively, you can remove the logic completely to recieve tokens through non-internal means.