RCT implementation tracking

LoCCS

August 25, 2017

1 Problem 1

Statement The first version of the full RingCT implementation is committed on 13/May/2016, in the commit "9b1afe" (for brevity, hash of commit is written in short format).

Reply The full RingCT implementation (a.k.a, the RingCT protocol in the codebase) is merged into master branch of the project in that commit. The core of the implementation involve mainly 2 functions, i.e., `rct::genRct` and `rct::verRct` (declared/defined in `src/ringct/{rctSigs.h,rctSigs.cpp}`), where

- `rct::genRct` is responsible for generating a MLSAG for a given ring and a range proof for the amount bound to addresses specified by the ring.
- `rct::verRct` is to verify the MLSAG and range proof from `rct::genRct`.

As seen, these core functions handle MLSAG generation/verification and range proof, which is only part of the job to do during constructing a transaction. Other subroutines for making a transaction include

- generate a random transaction key
- popuplate the input coins with their indices in the corresponding output transaction, etc
- derive addresses of destinations to send coins to
- ...

And the mix-ins ring is maded up randomly for `rct::genRct` for signing. After checking the codebase, I found that the transaction construction function `cryptonote::construct_tx_and_get_tx_key` hasn't integrated `rct::genRct` into its operation yet. Actually, `rct::genRct` and `rct::verRct` are not used in any other places up to commit "4fd01f".

2 Problem 2

Statement The first version of RingCT simple is committed on 9/Jul/2016, in the commit "4fd01f". The idea of RingCT simple is integrated in the process on 10/Jul/2016 in the commit "a4d4d6".

Reply

- In commit at "4fd01f", functions bound to the simple RingCT are implemented as `rct::genRctSimple` for signing and `rct::verRctSimple` for verifying. But they are not integrated into the transaction construction function `cryptonote::construct_tx_and_get_tx_key` yet, while the full RingCT has been employed.
- In commit at "a4d4d6" (one day after "4fd01f"), the simple RingCT functions are finally embedded into constructing transactions. Use the simple RingCT if all input coins are from previous RingCT transaction outputs (i.e., the amount mask value $a \neq 1$ for the corresponding amount commitment $a \cdot G + amount \cdot H$), and employ the full RingCT otherwise.

3 Problem 3

Statement The RingCT full and simple is defined as a type on 8/Aug/2016 in the commit "3ab2ab", i.e. `rv.type = RCTTypeFull` or `rv.type = RCTTypeSimple`. So, we might be able to run the RingCT full by set the type to `RCTTypeFull` in the current version of Monero.

Reply This commit just add a `type` to the signature object `rctSig` for future expansion, i.e., better scalability of the application. The transaction construction function `cryptonote::construct_tx_and_get_tx_key` remains almost unchanged as that in commit "a4d4d6". And the type of RingCT to make is determined systematically by that function other than manually, which means it cannot be set without modifying the code.

4 Observation

The criteria is updated at commit "a0925e" (27/Aug/2016 before the release verion 0.10.0 on 19/Sep/2016), for using which type of RingCT during transaction construction. As specified by the codebase and the message explaining the commit, the simple RingCT will be used in case of more than one coin to spend, and instead the full one is used in case of only one coin. And this criteria is kept up till now.

I found a post about the 3 transaction types (i.e., no/simple/full RingCT) in StackExchange. In that post, Lee Clagett, one of the contributors of the monero project, explains briefly that the simple RingCT is preferred over the full one primarily due to privacy concerns.

5 Summary

After its introduction, the simple RingCT is preferred over the full one, which is implemented in the codebase. With reference to the information above, I think the experiment to evaluate the original RCT and our RCT+ may be carried out as either of ways below.

- Scheme 1 (Different version of codebases):
  - For original RCT targeting multiple input coins, without modifying any code of the original project,
    - Evaluate the full RCT based on the codebase at commit "4fd01f"
    - Evaluate the simple RCT based on the codebase at commit "a4d4d6"
  - Evaluate RCT+ based on "a4d4d6"
- Scheme 2 (Latest codebase):
  - For original RCT targeting multiple input coins, based on the latest codebase, manually modify the criteria for choosing the type of RCT to use in the core function `cryptonote::construct_tx_and_get_tx_key` (actually just a few lines of code), and then evaluate the full RCT and the simple one respectively with slightly customized transaction construction function. Actually, the critical routine of constructing the transaction remains unchanged with respected to that in the codebase. It's just the rule to choose the simple RCT or the full RCT is modified slightly only, thus introducing no unwanted penalty affecting the performance evaluation.
  - Evaluate RCT+ based on the same codebase.