CONSENSYS

Diligence

RenVM Cryptanalysis Report

Prepared For:
Ren Project

Prepared By:
ConsenSys Diligence

Engagement Description

The ConsenSys Diligence team was contracted by Ren Project to conduct an analysis on the design and specification of RenVM, a secure multiparty computation element powering Ren Project's network.

Given the cryptographically heavy nature of the work, and the inability to accommodate work in its schedule at the time of contact, ConsenSys Diligence chose to conduct the review through two of its trusted independent partners:

- John G. Brainard
- ABDK Consulting

The outputs of both of these independent reviews are appended below in the order listed above.

Disclaimer

ConsenSys Diligence ("CD") typically receives compensation from one or more clients (the "Clients") for performing the analysis contained in these reports (the "Reports"). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within

this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.

Ren VM Cryptographic Review

John G. Brainard
Cryptography and Security Consultant

Draft 4
2020-05-13

Overview

This document is a cryptographic review of the secure multiparty computation protocol used to compute distributed ECDSA signatures in Ren VM.

Scope

The review is based on the whitepaper "*Ren VM Secure Multiparty Computation*", by Ross Pure and Zian-Loong Wang **[PW20]**. The latest version was distributed on April 29, 2020. The goal is to verify the theorems in the whitepaper and provide some assurance that the specified protocol has the security properties claimed in the paper.

Background

The theory of secure multiparty computation has been studied since 1986 **[Y86]**. Ran Canetti's 2000 paper entitled "*Security and Composition of Multi-party Cryptographic Protocols*" **[Can00]** is an important work in the field. In particular the paper provides a formal foundation for the assumption that if a protocol is built from secure components, and the components are properly composed, then the resulting protocol will be secure. Canetti's paper gives us the basis upon which to verify the claims in the whitepaper.

Document Review

General Notes

The whitepaper is very well written. It provides a very thorough description of the distributed signature computation and the cryptographic primitives on which the computation is based. The

mathematical foundation for each primitive is provided, along with necessary theorems and proofs.

The latest revision of the paper makes significant improvements over the original. In particular, moving some of the mathematical background and the more elaborate theorem proofs to the appendix, makes it easier to read and understand the entire document.

It might be useful to have a more detailed overview of the protocol at the start of the document. The overview could include a more traditional description of the protocol and the data exchanged. This would give the reader more context before introducing the mathematical models.

The paper gives no parameters for any of the cryptographic constructions used. While the math is valid for any choice of parameter, it would be useful to suggest a set of parameters that might actually be used in implementing the protocol. This would allow for better comparison with other such protocols.

Notation and Definitions

The whitepaper uses standard computer science notation for the terms and operations defined in the paper. The definitions explain the mathematical concepts used in building the protocol and provide a clear basis for understanding the theorems.

Secret Sharing (Section 2.4)

The secret sharing scheme is a straightforward usage of Shamir's scheme. Theorem 3 asserts the soundness of the scheme. The proof, in Appendix C, considers three cases with different numbers of shares relative to the threshold, and computes the output probability for each.

Discrete Log Assumption (Section 2.5)

The discrete log assumption is the standard for DSA implementations. Stating the assumption using Turing machines is fairly novel, but the logic seems sound.

Security Model (Section 3)

The security model defines both "real-life" and "ideal" cases. This model is consistent with Canetti's work. Both cases are described well and the corresponding functions (ADVR and EXEC) are well defined and consistent.

Definition 7 asserts a “t-limited adversary.” The meaning of t-limited can be inferred from context, but a brief definition would be useful.

Definition 9 identifies the key security requirement for the protocol and is consistent with a similar definition in Canetti’s paper.

Public Key Encryption (Section 4.1)

The definition of PKE is a bit terse, but since the concept should be familiar to any reader, this is not really an issue.

Zero Knowledge Proofs (Section 4.2)

The algorithms needed for a zero knowledge proof (setup, prove, verify, and simulate) are defined clearly. The security properties required: completeness, perfect zero-knowledge, and soundness are specified in Definitions 10-12. The definitions appear both clear and correct.

Consensus Algorithm (Section 4.3)

The consensus algorithm specified here seems pretty straightforward. The most novel aspect is the use of zero knowledge proofs instead of commitments or other lighter-weight constructions. This may add some computational cost, but it also may give better assurances with fewer rounds of communication. A more thorough discussion of the tradeoffs involved and, possibly, a comparison with other algorithms might be good.

The Consensus algorithm is used as a basis for the Biased RNG subprotocol in Section 6.2.1.

ECDSA Algorithm (Section 5)

The description of the DSA algorithm is standard as are the modifications required for use with elliptic curves.

The algorithm is compared with the well-known GJKR algorithm [GJKR01]. The differences are the changes required to use ECDSA instead of DSA, and the composition of the protocol from subprotocols. The subprotocols are described in Section 6.

Open (Section 6)

The Open subprotocol reveals the result of a secret sharing. Three different variants of this subprotocol are presented: Share Revealing Open, Share Hiding Open, and Directed Open.

Reed-Solomon Decoding (Section 6.1.1)

RS Decoding allows the recovery of a message in the presence of errors. In this protocol it is used to reconstruct the secret when some shares are missing or corrupted. RS encoding and decoding is well understood and a good choice for this application. The RS parameters (RS(255,233)) are determined from the number of parties in a straightforward way. It might be useful to have an example from a typical application, showing the number of parties, the threshold, and the RS parameters.

Share Revealing Open (Section 6.1.2)

This subprotocol is relatively simple: each party first broadcasts its inputs share, then reconstructs the secret (using RS decoding) and broadcasts the result along with the reconstructed shares. Theorem 6 asserts the security of the subprotocol. The proof is straightforward.

Share Hiding Open (Section 6.1.3)

This subprotocol is similar to the Share Revealing Open, but adds a Random Zero Sharing (see Section 6.2.3) used to mask the input shares and only broadcasts the recovered secret. Theorem 7 asserts the security of the subprotocol. The proof, in Appendix D, is very detailed, and the logic is clear.

Directed Open (Section 6.1.4)

The Directed Open subprotocol reveals the secret to a single party, instead of all parties. It has both share-revealing and share-hiding versions. This subprotocol is just a subset of the Share Hiding Open and Share Revealing Open subprotocols. Theorems 8 and 9 assert the security of both variants of the subprotocol. The proof of security for each is inherited from the prior subprotocols.

Biased RNG (Section 6.2.1)

The Biased Random Number Generation subprotocol, designated π_{BRNG}, uses the consensus algorithm to create a random number that all of the parties in the protocol agree to. Theorem 10

asserts that the subprotocol has the properties of agreement, randomness, and secrecy against an adversary who can corrupt less than the threshold number of parties. The proof builds nicely on earlier Theorems and appears sound.

Unbiased RNG (Section 6.2.2)

π_{RNG}, the Unbiased Random Number Generation subprotocol, is built from π_{BRNG}. π_{BRNG} is invoked k times (where k is the threshold value) and the results are used to construct a polynomial which is shared by a random sharing.

Theorem 11 asserts that the subprotocol is secure against an adversary who can corrupt less than the threshold number of parties. The proof, in Appendix E, uses a simulator to demonstrate that the results of the subprotocol are uniform, even in the presence of a rushing adversary.

The term “rushing adversary” is used without explanation. A reference to Fehr and Yuan’s paper [FY19] might be helpful here.

Random Zero Generation (Section 6.2.3)

The Random Zero Generation subprotocol, π_{RZG}, is the splitting of the zero value between parties, using a random sharing. Theorem 12 asserts that the subprotocol is secure against an adversary who can corrupt less than k-1 parties, where k is the threshold. The lower threshold for the adversary is because the fixed value being distributed is known. The proof is omitted, since the logic is identical to that for π_{RNG}.

Random Keypair Generation (Section 6.2.4)

The Random Keypair Generation subprotocol, π_{RKPG}, extends π_{RNG} to generate an ECDSA key pair. Each party receives a share of the private key and the entire public key.

Theorem 13 asserts that the subprotocol is secure against a passive adversary. The proof is simple since all the information needed is provided by the trusted parties.

Theorem 14 asserts that any computation based on a number of shares less than the threshold has a negligible probability of determining the secret key, assuming the hardness of discrete log. The proof assumes that an algorithm to compute the secret key exists, then shows that it implies the ability to compute discrete logs.

Finally, the subprotocol is enhanced to be secure against an active attacker by adding a zero-knowledge proof of the correctness of each private key share. This allows the detection of incorrect values and prevents them from corrupting the result.

Multiply and Open (Section 7.1)

The Multiply and Open protocol, π_{MO}, produces and shares the product of two values from the shares of the individual values. The math is straightforward since, for any multiplicative sharing, the product of two shares is a share of the product. π_{SO} is used to share the result.

Theorem 15 asserts that the protocol is secure for appropriate numbers of shares.

Inversion (Section 7.2)

The Inversion protocol, π_{INV}, computes and shares the multiplicative inverse of a value, given the shares of that value. π_{RNG} is used to mask the share values.

Theorem 16 asserts that the protocol is secure under the same conditions as π_{MO}.

Sign (Section 8)

π_{SIGN} is the full ECDSA signature protocol. It is composed from π_{RKPG}, π_{INV}, and π_{MO}. π_{RKPG} is used to create shares of a random point and its product with the curve generator. π_{INV} inverts the shares of the random point. Finally, π_{MO} computes and distributes the product of the inverted shares with the message plus the private key.

Theorem 17 asserts that the signature protocol is secure, under the conditions inherited from the subprotocols.

Conclusions

The document “RenVM Secure Multiparty Computation” is an impressive work. The subprotocols are clearly and logically defined. Each subprotocol is given a solid mathematical foundation. The full protocol is composed from the subprotocols in a way that is easy to understand and maps naturally to the standard description of the ECDSA algorithm. The

composition of the protocol adheres to the principles of Canetti, allowing the final protocol to inherit the security properties demonstrated for the subprotocols.

The only shortcoming observed here is a lack of “real world” examples of the usage of the protocol, with parameter values. This could be addressed in a later version of this document, or in a companion document, such as an implementer’s guide.

References

[Can00] Canetti, R. “*Security and Composition of Multiparty Cryptographic Protocols*”, J. Cryptology13, 143–202 (2000). https://doi.org/10.1007/s001459910006

[FY19] Fehr and Yuan, “*Towards optimal robust secret sharing with security against a rushing adversary*”, Cryptology ePrint Archive, Report 2019/246, 2019, https://eprint.iacr.org/2019/246

[GJKR01] Gennaro, Jarecki, Krawczyk, and Rabin, “*Robust Threshold DSS SIgnatures*”, Information and Computation, Volume 164, Issue 1, 2001

[PW20] Pure and Wang, “Ren VM Secure Multiparty Computation”, April 2020

[Y86] A. Yao, “*How to generate and exchange secrets*”, In *Proc. 27 Annual Symp. on Foundations of Computer Science (FOCS)*, pages 162-167. IEEE, 1986

RenVM review

ABDK Consulting

April 17, 2020

Contents

1 Introduction

This document is a review of "RenVM Secure Multiparty Computation" authored by Ross Pure and Zian-Loong Wang. We have reviewed a version of this paper obtained from a private repo[1].

2 Original paper

2.1 Main results

The paper presents a protocol to compute the ECDSA signature in the framework of secure multiparty computation (secure MPC); this is also known as threshold ECDSA. The authors claim the security with $t < \min(n/2 - k, k)$ corrupted parties where at least k out of n parties must cooperate to sign. The mathematical description of the protocol and some proofs of security are given. Complexity estimates and benchmarks are not reported.

2.2 Paper structure

The paper starts with a detailed introduction into probability theory and statistics in order to rigorously define the security of the main protocol (Section 2). Then they briefly define the security of the protocol and describe some non-MPC primitives used (Sections 3 and 4). The secure MPC subprotocols (gadgets) follow in Section 5, whereas the main protocol is given in Sections 6,7.

3 Audit

3.1 Related work

The problem of multiparty (threshold) computation of ECDSA can be traced back to the 1990s when the same problem was studied for DSA (DSS). It was demonstrated that it is possible to collectively sign so that any less than t parties can not sign [GJK+96]. However, this requires that at least $2t$ parties share the key and, again, at least $2t$ parties must participate in the protocol (the number of shares can of course be higher).

The need for more robust threshold schemes was frequently asserted; moreover the existing schemes were not suitable for 2- and 3-party computation [MR01]. Recently, the first schemes that provide almost optimal security were proposed using the idea of zero-knowledge proofs and homomorphic encryption [GGN16; GG18; LN18].

3.2 Overall impression

We have found that the protocol is heavily based on the 1996 threshold DSS protocol [GJK+96]. However, the protocol part is described vaguely in many places, and the actual differences to[GJK+96] are unclear. The report should be restructured in order for

[1] https://github.com/renproject/mpc-specification/blob/master/z0_spec.pdf

the new material to be analyzed properly. We summarize the main generic issues of the report in the following paragraphs.

State of the art The report does not discuss the state of the art in the threshold/MPC ECDSA signing. It should, at least briefly, point to the main results in the area, compare the suggested protocol to them, discuss their limitations and advantages, explain why this particular protocol has been chosen. Currently it is difficult to judge the decisions taken by the authors as it is hard to figure out what has been borrowed from where.

Missing references The paper provides little to no references not only to the related work but also to generic cryptography and mathematical constructions, which are heavily used (from statistical arguments to public key crypto). All constructions taken from third-party papers [BB89; GJK+96] should be referenced except for common knowledge results. The statistics part should contain references to some textbooks if the authors consider all the proofs and definitions necessary, though from our point of view most can be skipped, see below.

Novelty is unclear Even though we found that the protocol is heavily based on [GJK+96], there can be some subtle places which differ. The authors should makes these parts explicit and clearly distinguish between the original and third-party results.

Excessive formalism A big portion of the paper is devoted to basic definitions and results in probability theory and statistics. The formalism in the introduction is excessive and unnecessary for the high-level protocols like MPC-ECDSA, as the resulting report is not self-containing anyway. Most results and their proofs in Section 2 are standard and do not have to appear in a protocol paper. The authors should provide just the basic non-trivial definitions of MPC security and explain what the security means from the statistical point of view. A more curious reader should be referred to one of classical textbooks.

Features are missing The authors should provide a summary of the protocol in a regular form: what kind of security it provides, which features it supports, does it satisfy various MPC-related requirements (e.g. universal composability). The same applies to subprotocols.

Adversarial model is unclear The authors mention that they consider passive adversaries, active adversaries, computationally bounded adversaries – but often the type of adversary is defined only vaguely. We recommend to describe the adversarial model explicitly in the beginning. The common types of adversaries are semi-honest, malicious, covert, etc. [Lin20].

Security model is unclear The authors occasionally mention that they aim for the perfect security. However, it is not clear (and probably untrue and unnecessary) that all the considered protocols have perfect security.

Parameters are unset Parameters needed for secure computation are not set (e.g. public-key crypto scheme and its details).

Vague protocol description In contrast to Section 2, the protocols are defined vaguely. Authors should use recent MPC papers as an example of real-ideal functionality presentation [GG18].

3.3 Concrete issues

Section 2:

1. Theorem 3, Section 2.4 – the statement is standard and can be found in many presentations on Shamir secret sharing.

2. Theorem 5, Section 2.4 – the statement is a standard result on the Lagrange interpolation.

3. Definition 1, Section 2.5 – it is uncommon to use Turing machines in security definitions. Usually we talk about adversaries of various computing and/or corrupting power and distinguish between uniform and non-uniform adversaries. Also if T takes G as input, then G should denote a group rather than be a group.

Section 3:

1. Definition 12 is unclear. Also "t-limited" adversary is undefined.

Section 4:

1. Definition 15 defines the zero-knowledge property of a proof system but does not include soundness. As the latter is important property in protocols employing ZK proofs, there must be a remark on the soundness feature.

2. Section 4.3 – the authors mention that they use a consensus algorithm for some protocols. It is not clear from the rest of the paper where and why such algorithms are required. In general, consensus algorithms are used whenever a secure broadcast is not available. The paper should provide some arguments why and when consensus algorithms are needed in the application.

3. Section 4.3, the $\rho_{RNG}^{n,k}$ protocol is defined very informally in a big contrast with section 2. It is unclear how the shares are created, what kind of ZK proof is involved and so on. No concrete details are provided. The "verify" function is undefined.

4. Section 4.3 "Recall that the two key properties that definition 14 captures are secrecy and correctness." – there are no such properties in Definition 14. The rest of the paragraph is vague and its purpose in unclear.

Section 5:

1. "It is clear that the security model trivially applies when a protocol is local" – it is not clear since the security model is not well-defined.

2. Section 5.1 – "almost certainly" is vague and needs reformulating. The rest of the paragraph is written in conditional which confuses the reader as it is not clear what has been decided eventually.

3. Section 5.1 – references to the related work (on open protocols and Reed-Solomon enhancements) are missing.

4. Section 5.1 is vague. Wordings like "often", "may be" etc. make the protocol ill-defined.

5. Theorem 6 mentions "t-secure" computation without defining it. One may only guess that this means corrupting up to t parties.

6. Section 5.1 – the usage of Reed-Solomon codes and decoding algorithms is not fully defined.

7. Section 5.1, Remark 3 – it is unclear if this way is taken or not.

8. Section 5.1, Remark 4 – "When $k = 2n/3$, which occurs during multiplication of two k-sharings when $k = n/3$" – something is misspelled here. Same paragraph, "We could achieve a more relaxed $t < n/3$ requirement" – so is this used or not? What kind of ZK proofs is needed for that? What about soundness?

9. Theorem 7 – though the intuition behind the proof makes sense, we are not convinced that the variables X and Y are the "key randomness". Probably this proof can be acquired from another paper.

10. Section 5.1.1 – it is unclear if the directed versions of the protocol are used or not.

11. Section 5.2.1 – "it's just the sharing that has been biased" – this sentence is unclear. The same paragraph also misses references.

12. Section 5.2.1 – "our first biased RNG protocol" – it is unclear which one you mean.

13. Section 5.2.1 – the protocol Biased-RNG is not fully defined in either (real/ideal) world, and it is unclear how exactly the RNG protocol is used here. i.e. how the subset is chosen and what the role of consensus is. Consider adding a formal description.

14. Theorem 10 – there should be a simpler argument that secrecy holds that uses the fact that $t < k$. The authors should also provide some intuition why j is taken as a point where the polynomial is computed. Can there be other variants?

15. Section 5.3 – as the protocol and proof for π_{RZG} is almost the same as π_{RNG}, can it be one protocol and one theorem?

16. Section 5.4 – since ECDSA is discussed, the additive notation for group operation should be used rather than the multiplicative one.

17. Section 5.4, page 32 – the authors should provide some candidates for commitment scheme and ZK proofs alongside with the set of criteria. Currently only the perfect zero-knowledgeness and hiding are required, but obviously the ZK proof system must be also sound and complete.

Section 6:

1. The introduction is quite confusing. It should simply state which protocol is constructed and what its features are.
2. Section 6.2 should have comparison to Beaver triples [Bea91].
3. The proof of Theorem 17 is formally missing.

References

[BB89] Judit Bar-Ilan and Donald Beaver. "Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction". In: *PODC*. ACM, 1989, pp. 201–209 (cit. on p. 3).

[Bea91] Donald Beaver. "Efficient Multiparty Protocols Using Circuit Randomization". In: *CRYPTO*. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 420–432 (cit. on p. 6).

[GG18] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: *ACM Conference on Computer and Communications Security*. ACM, 2018, pp. 1179–1194 (cit. on pp. 2, 4).

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. "Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security". In: *ACNS*. Vol. 9696. Lecture Notes in Computer Science. Springer, 2016, pp. 156–174 (cit. on p. 2).

[GJK+96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, et al. "Robust Threshold DSS Signatures". In: *EUROCRYPT*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 354–371 (cit. on pp. 2, 3).

[Lin20] Yehuda Lindell. "Secure Multiparty Computation (MPC)". In: *IACR Cryptology ePrint Archive* 2020 (2020), p. 300 (cit. on p. 3).

[LN18] Yehuda Lindell and Ariel Nof. "Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody". In: *ACM Conference on Computer and Communications Security*. ACM, 2018, pp. 1837–1854 (cit. on p. 2).

[MR01] Philip D. MacKenzie and Michael K. Reiter. "Two-Party Generation of DSA Signatures". In: *CRYPTO*. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 137–154 (cit. on p. 2).