22. Nachwuchswissenschaftler*innenkonferenz (NWK)

https://doi.org/10.52825/ocp.v2i.133

Published: 15 Dec 2022

Verifying the Elliptic Curve Verifiable Random Function Secp256r1 on Blockchain

Nomana Ayesha Majeed[1] and Alex Kemloh Kouyem[1]

[1] Fakultät Angewandte Computer und Biowissenschaften, Hochschule Mittweida

[*] Corresponding author: majeed@hs-mittweida.de

Abstract. Randomness is a critical issue in peer-to-peer networks because random numbers allow us to fairly select the candidates, resolve the lotteries, select block producers, etc. The difficulty is that each participant wants to verify that the random number was randomly generated, this led to the invention of verifiable random functions (VRF). The VRF is a pseudo-random function that provides a solution to blockchain-based random number generation.

This paper focuses on the implementation of an elliptic curve-based VRF introduced by NIST called ECVRF-secp256r1 in Solidity. The algorithm verifies the randomness on-chain, tests the gas consumption at each stage, and compares it to the ECVRF-secp256k1.

1. Introduction

Providing randomness to the blockchain without compromising the decentralization is critical due to the deterministic nature of the blockchain. Four important directions namely cost, time, randomness, and security need to be considered while generating a random number. There are several methods such as commit and reveal, blockchain as a source of randomness, Oracle pattern, etc. But none of these methods ensure all four domains, this leads to the invention of verifiable random function.

2. Technical background

2.1 Blockchain

The Blockchain is a distributed immutable append-only database that is shared among the nodes of a peer-to-peer computer network. Ethereum is a decentralized blockchain platform that securely executes and verifies application code using smart contracts [1]. It offers a flexible platform to build decentralized applications using Solidity and EVM.

2.2. Secp256k1 and Secp256r1 curve [2]

Secp256k1 curve is based on the Koblitz curve whereas the Secp256r1 curve is a random curve introduced by NSA. Regarding secp256r1, it is assumed that the randomly selected parameters are more secure but there is a concern that random coefficients can provide the black door since it is almost impossible to prove that they are purely random. Parameters in the secp256k1 curve are chosen with relatively high rigidity therefore, it is considered as more efficient curve.

2.3 Elliptic curve VRF [3]

VRF is a cryptographic primitive that maps input to a verifiable pseudorandom output. For a key pair (pk, sk) and an input x, a VRF produces a unique pseudorandom verifiable output and a non-interactive proof of correctness. It satisfies three properties called provability, pseudorandomness, and trusted uniqueness. Additionally, elliptic curves have an advantage in terms of small security key size over RSA.

3. Our Contribution

We implemented an elliptic curve based VRF [4] for both secp256r1 and secp256k1 curves in JavaScript and Solidity. We executed the proof off-chain using JavaScript because transmitting the proof on-chain would reveal the secret key. However, the smart contract verifies the proof on-chain using the verify function. We also implemented a fast way to verify proofs by performing scalar multiplications off-chain, since adding and multiplying elliptic curve points on the blockchain is expensive.

We tested the gas consumption based on 10 randomly selected secret keys and seeds of different lengths for both curves at each stage. The following table summarizes the min, max, and average gas consumption:

Table 1. Average gas consumption for secp256k1 and secp256r1 curve

Methods (Solc 0.6.12, optimizer: true)	secp256r1				secp256k1			
	min	max	avg	avg cost	min	max	avg	avg cost
fastVerify	55035	197198	76550	$0.21	65501	203857	99781	$0.27
verify	1489351	1749433	1615921	$4.47	1529800	1786976	1634355	$4.51
proofToHash	23564	23600	23596	$0.06	23564	23600	23597	$0.06
computeFastVerifyParams	1487122	1747195	1613680	$4.46	1527608	1784779	1632151	$4.51
decodeProof	50523	50568	50548	$0.14	56837	56894	56872	$0.16
hashAndTryIncrement	47626	189785	69141	$0.19	58104	196447	92374	$0.25
hashPoints	28465	28513	28499	$0.08	28453	28513	28497	$0.08

According to the table 1, we concluded that secpr1 consumes slightly less gas than secpk1 (on average < 23.231 gas) [5]. By using the “computeFastVerifyParams” we can save on average ~1613680 gas.

To identify the difference between functions fastVerify, verify, and hashAndTryIncrement, we have analyzed these functions against the test vectors as given below:

Table 2. Average gas consumption for secp256k1 and secp256r1 curve using different seeds and keys

Secret keys:		c9afa9d845ba75166b5c215767b1d6934e50c3db36e89b127b8a622b120f6721		817f508d2d36aaa7ca077d1fd3c27d71075aa979c59a9b75fc4a29b8cd27f3e0	
seed	methods	secp256r1	secp256k1	secp256r1	secp256k1
' '	fastVerify	102333	99614	55035	99602
	verify	1651202	1644866	1563218	1630763
	hashAndTryIncrement	94912 (3 rounds)	92229 (2 rounds)	47626 (1 round)	92229 (2 rounds)
'test'	fastVerify	55229	65723	102519	99867
	verify	1530011	1601713	1636950	1661479
	hashAndTryIncrement	47820 (1 round)	58314 (1 round)	95106 (3 rounds)	92455 (2 rounds)
'optimization'	fastVerify	78927	65834	78991	65806
	verify	1651491	1655153	1672512	1555690
	hashAndTryIncrement	71554 (2 rounds)	58410 (1 round)	71570 (2 rounds)	58394 (1 rounds)
'VRF is pretty amazing'	fastVerify	79118	134231	55464	65942
	verify	1634215	1688928	1623232	1584749
	hashAndTryIncrement	71697 (2 round)	126810 (3 rounds)	48043 (2 rounds)	58521 (1 round)
'Example of EC-DSA with an-sip256r1 and SHA-256'	fastVerify	127557	66600	56110	66588
	verify	1749433	1614904	1587918	1633671
	hashAndTryIncrement	120151 (4 rounds)	59179 (1 round)	48701 (1 round)	59179 (1 round)

The function hashAndTryIncrement used to generate a valid curve point for both fastVerify and verify consumes more than 80% of the total amount of gas. Some combinations (public key + seed) requires the algorithm to perform several rounds to search for a valid curve point, this causes an increase in gas consumption since each new round cost ~20,000 - 40,000 gas. Additionally, hashAndTryIncrement consumes much gas therefore, the question is whether we can also perform this function off-chain? This will allow us to save ~69,141 gas for the secp256r1 curve and ~92,374 gas for the secp256k1 curve.

4. Conclusion

Applications using VRF must ensure that the key pair is generated correctly using good randomness. In addition, implementers should be careful that the proof does not obscure the seed. Anyone who knows the public key and proof can use an off-chain dictionary attack to search for the seed by checking guesses for the seed with VRF_verify. Chainlink [6] offers a random number generator using VRF with secp256k1 curve but the cost is extremely high. Implementing VRF with secp256r1 would slightly reduce the costs but performing scalar multiplications off-chain will drop the costs dramatically.

Data availability statement

The underlying data is available upon request from the authors.

Competing interests

The authors declare that they have no conflicts of interest.

References

1. Ethereum.org (2022), “ethereum.org,“. https://ethereum.org/en/what-is-ethereum/. [Accessed 14-03-2022].
2. Brown, D. R. (2010). Sec 2: Recommended elliptic curve domain parameters. Standars for Efficient Cryptography
3. Micali, S., Rabin, M., & Vadhan, S. (1999, October). Verifiable random functions. In 40th annual sym-posium on foundations of computer science (cat. No. 99CB37039) (pp. 120-130). IEEE.
4. S. G. L. R. J. V. Dimitrios Papadopoulos (2021): "Verifiable Random Funtions (VRFs),". https://datatracker.ietf.org/doc/pdf/draft-irtf-cfrg-vrf-06. [Accessed 05-01-2022]
5. Cao, M. (2021): “Announcing our Verifiable Random Function (VRF) library in Solidity”. https://medium.com/witnet/announcing-our-verifiable-random-function-vrf-library-in-solidity-c847edf123f7. [Accessed 12-02-2022]
6. Chainlink Developers: “Introduction to chainlink VRF”. https://docs.chain.link/docs/chainlink-vrf/ [Accessed 19-04-2022]