

# Intuitive Interpretation of Non-Interactive Zero-Knowledge Cryptography

A jargon-free approach to understanding zk-SNARKs and zk-STARKs

Avneet Singh  
Interplanetary Company UG  
[sshmatrix@proton.me](mailto:sshmatrix@proton.me)

## ABSTRACT

zk-SNARKs and zk-STARKs are relatively new concepts in cryptography, yet they are being touted as the next forefront in modern and future crypto tech. In blockchain space specifically, there is great interest in these subfields in the context of zk-Rollups to Layer 1 blockchains such as Ethereum, or as standalone decentralised ledgers with high rates of transactions per second (TPS), e.g. Aztec Network (zk-STARK), zkSync, Loopring, ZCash (zk-SNARKs) etc. Despite their great importance in cryptography, it is unfortunately difficult to understand zk-SNARKs and zk-STARKs due to limited literature and conceivably difficult mathematics conveyed through intensive jargon. This paper is an attempt to introduce zero-knowledge (zk) cryptography in an intuitive manner to garden-variety mathematicians, physicists, curious blockchain developers and perhaps even cryptographers.

## INTRODUCTION

Zero-knowledge cryptography is presumably the next natural stage in cryptography's evolution toward post-quantum era. zk-SNARKs and zk-STARKs are specific implementations of zero-knowledge cryptography that are widely considered the most promising path toward post-quantum security. In order to deeply understand zero-knowledge cryptography, one must at least understand the current generation cryptographic systems such as RSA (Rivest-Shamir-Adleman) and Elliptic Curve Cryptography (ECC); this is admittedly a challenging task since the mathematics of such protocols is rather tedious and it only gets exponentially worse as one ventures into zero-knowledge protocols. Despite these challenges, it is nonetheless easier to understand at least the philosophy and intuition behind zero-knowledge protocols using the famous Alibaba Cave example [1] without requiring any prerequisite knowledge of RSA or ECC. We leave this as an exercise for the reader. In this paper, we will attempt to delve into the practical implementation of zero-knowledge protocols while retaining an intuitive understanding of the underlying mathematical processes.

Through the course of this paper, we will leave additional comments in a grey box targeted at physicists, mathematicians and developers. These comments are anecdotes, comparisons or similarities noted across different fields that may help readers develop an intuitive understanding.

Cryptographic protocols at their core are motivated by the need to prove access to some information without necessarily revealing a part or the entirety of said information. In mathematical terms, there are several ways of achieving this functionality from an intelligently designed system.

## PRIME NUMBERS

Cryptographers realised back in the day that prime numbers were one such system

that could provide the desired functionality. For instance, consider two sufficiently large prime numbers  $a = 53781811$  and  $b = 23252729$ , and their even larger product  $a \times b = 1250573876312219$ . The product  $1250573876312219$  is a relatively difficult number to prime factorise back to  $a$  and  $b$  if both  $a$  and  $b$  are unknown. However, if either one of the two prime factors ( $a$  or  $b$ ) are known, then it is straightforward to calculate the other unknown prime factor by simple division. To intuitively understand this system further, let's break it down into its principle components: we took a set of very large prime numbers of which  $a$  and  $b$  are members and defined an operation of multiplication<sup>1</sup> on the members of the set; such a finite field is called a Galois field. Additionally, we note that the product operation  $\times$  is difficult to invert unless one of the two numbers is known; this kind of a system is usually called a trapdoor.

In abstract mathematics, such a system is called a Group and the study of groups is called Group theory; a group is defined by a set of elements (called a Field, e.g. large prime numbers) along with the set of permitted operations between those elements (e.g. multiplication). The configuration of any trapdoor system is such that the permitted operations defined on the elements of the group are difficult to invert.

This trapdoor feature of our chosen system is essentially the backbone of all present day cryptography.

## RSA CRYPTOGRAPHY

RSA protocol is one of the simplest implementations of the trapdoor feature which results in a keypair system – a public key and a private/secret key – typically utilised in conjunction for encrypting and decrypting information. The premise of the RSA protocol essentially lies in setting, i) public key as the qualitative equivalent of the product  $a \times b$ , and ii) private key as the qualitative equivalent of either  $a$  or  $b$ , where  $a$  and  $b$  are restricted to a finite field of prime numbers (denoted by  $F_p$ ). The security of such a system is encoded in the difficulty of prime factorising the product. It is unimportant to know the precise details of the protocol implementation in context of this paper. In nutshell, the RSA algorithm requires solving for the coefficients  $(x, y)$  of Bezout's Identity ( $xa + yb = 1$ ) using extended Euclidean algorithm [2] and employing modular arithmetic to wrap integer numbers when they fall outside the range of a separately defined composite finite field  $F_p$ .

Modular arithmetic is quite simply equivalent to adding the modulo '%' operation over some prime number  $p$  to all group operations  $\bullet$  (i.e.  $\bullet = +, -, \times$  or  $/$ ), such that  $a \bullet b \rightarrow a \bullet b \% p$ . Modular arithmetic is regular arithmetic 'wrapped' by the modulo operator such that results of all group operations lie within  $0$  and  $p$ .

## ROBUSTNESS OF RSA

While RSA is sufficiently safe to use today, it's safety will decrease over time as computational capacity of human civilisation increases. This is because the fastest and – arguably – maximally efficient algorithms capable of inverting the  $a \times b$  product are iterative by construction and rely on optimised brute-forcing; Quadratic Sieve [3] and General Number Field Sieve [4] are two

<sup>1</sup>in addition to implicitly defining the operation of addition '+' and its inverse subtraction '-'

such well-known methods. With the advent of quantum computers, RSA algorithm's security will be definitively compromised; this is the so called SNDL problem (Save-Now-Decrypt-Later) facing the cryptography community today [5]. The core issue at hand here is that finite prime fields and the generic operation of multiplication on them – irrespective of the largeness of its elements – does not possess sufficient difficulty if the exploiter has relatively large computational resources at hand. In order to design better cryptosystems that are secure against brute-forcing, one must come with a new group with a better choice of finite field and a preferably a harder group operation which is resistant to such an attack.

## DIFFIE-HELLMAN KEY EXCHANGE

In very few lines, let us discuss the Diffie-Hellman Key Sharing Algorithm. Diffie-Hellman is another cryptosystem which in some sense is the intermediate step between RSA and ECC since it introduces finite fields over groups other than itself. In simple terms, this means that the group operation of product  $\times$  remains the same but the group elements are now that of a finite cyclic group [6]. The non-invertible term securing the protocol in this case is of the form<sup>2</sup>  $G^K \% p$ , instead of a simple product between two large prime numbers. Breaking the Diffie-Hellman method is thereby different from RSA since it involves inverting  $G^K \% p$  instead of  $a \times b$ ; this is known as the discrete logarithm problem.

Note that the term  $G^K$  is better not interpreted as a literal exponent although a lot of literature uses this terminology.  $G^K$  is a representation for a group operation that 'operates on or with itself'  $K$  number of times. In context of cyclic groups,  $G$  is the generator element of the group (of order  $N$ ), i.e. the element from which all other elements can be derived by iteratively performing the  $G^K$  operation for  $K = 1, 2, 3 \dots N$ .

## ELLIPTIC CURVE CRYPTOGRAPHY

The fundamental issue that leads to the breakdown of RSA algorithm is that the prime number field and the operation of multiplication is not complex enough in the face of large computational power. The effort to improve on this problem led to the advent of Elliptic Curve Cryptography (ECC). ECC functions by fixing the core illnesses in RSA, i.e. it proposes – similar to Diffie-Hellman – that instead of operating on the prime number field itself, we operate on another field on which we define a newer set of operations that are much harder to invert. The premise of ECC thus lies in the introduction of finite prime fields over elliptic curves, a step up from finite cyclic groups used in Diffie-Hellman [7]. In context of cryptography, an Elliptic curve is simply the following relation between a finite prime field  $x$  and composite field  $y$ ,

$$y^2 = x^3 + px + q,$$

$$\text{given } 4p^3 + 27q^2 \neq 0 \quad (1)$$

Gigabrain realised few decades ago that the elliptic curves have some very cool properties in context of invertible binary operations. To begin with,

1. The elliptic curve is symmetric across the  $x$ -axis; this is a desirable property since it ensures the existence of both  $y$  and  $-y$  in our group. More importantly,

---

<sup>2</sup> $G$  is called the group generator (integer),  $K$  is private key (integer) and  $p$  is a prime number

2. For any two points  $a$  and  $b$  on the elliptic curve, a line through them intersects the elliptic curve again at most – but not necessarily – 1 more point, denoted by  $c$ ; the blue line in figure 1 is one such example.

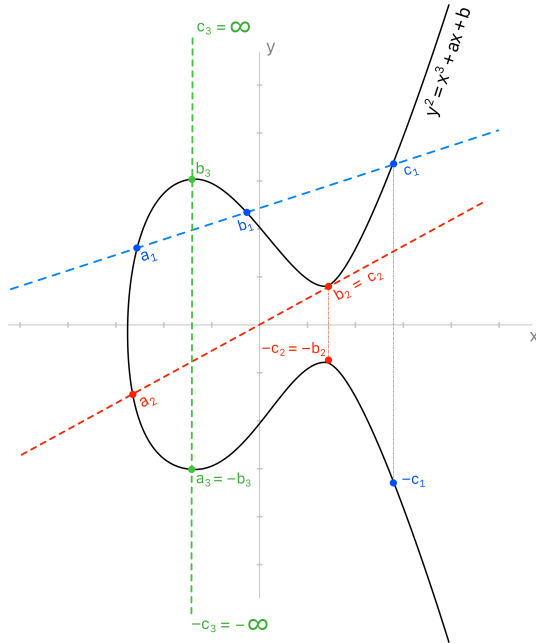


Figure 1: Elliptic curve over a continuous infinite field, aka a Lie group. We ignore the finite field constraint in this figure for simplicity.

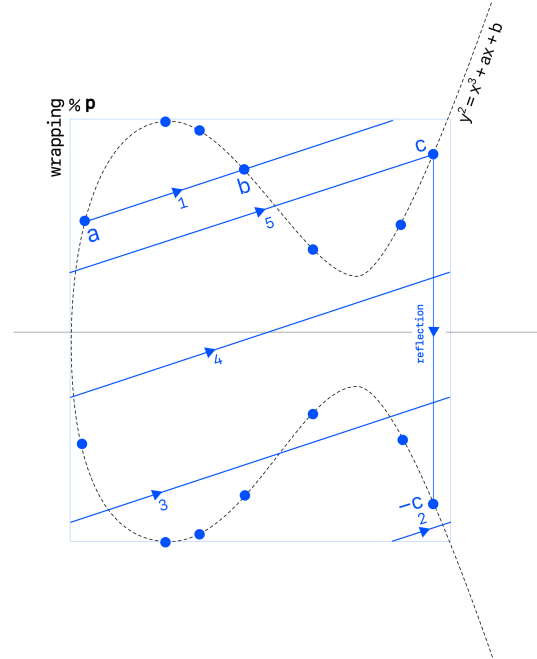


Figure 2: Elliptic curve over a finite prime field modulo  $p$ ; the effect of the modulo operator  $\%$  is to "wrap" the out-of-range values to within range. In the above example, it takes 5 'wrappings' and 1 final reflection to calculate  $-c$ .

Typically,  $x$  is taken as the independent finite prime field and  $y$  is the so-called 'elliptic curve (evaluation) over finite field';  $y$  are the elements of our desired new group. Ignoring the finite field constraint for a moment, the elliptic curve is shown in figure 1 when  $x$  is a continuous real variable. Our task now is to come up a hard-to-invert group operation on the continuous elements  $y$  of our new group; we will reinstate the finite prime field constraint afterwards. Note that it is possible for the line to not intersect the curve for the third time at all, and these singularities are in fact necessary for our group to function. Consider, the green line in figure 1; note that all vertical lines intersect the curve at exactly two points. It is also possible for a tangent to the curve to intersect the curve at exactly two points; see one such example (the red line) in figure 1. The aforementioned gigabrain soon realised that one can use this intersectional property of a line with the elliptic curve (aka properties 1. and 2.) as the new group operation  $\oplus$  such that

$$a \oplus b = -c \quad (2)$$

This is the same as saying that  $\oplus$  operation between  $a$  and  $b$  is equivalent to taking reflection ( $-$ ) of the third intersection point  $c$  across  $x$ -axis. Most literature phrases this  $\oplus$  operation as the new "addition" operation on the group elements of  $y$ . However, we noted before that the third intersection  $c$  doesn't exist for all pairs  $a$  and  $b$ ; in jargon-speak, we say that 'the group is

not closed under the  $\oplus$  operation'. We can solve this closeness problem by explicitly including the infinity element  $\infty$  to the group. Once we have access to  $\infty$ , all intersections similar to the green line are assumed to intersect the curve again at  $c = \infty$ , such that

$$a \oplus \infty = a \quad (3)$$

Note that  $\infty$  acts like the identity element  $\hat{1}$  of our new group since it satisfies  $a \oplus \hat{1} = a$ . Lastly, the third case of the red line is not difficult since it is simply a special case of the blue line when  $b = c$ ; another important tangent case arises when  $a = b$ , such that  $a \oplus a = c$ . Now we have all the ingredients required to construct a robust inverse problem on our new group  $y$  with the permitted operation  $\oplus$ . We reinstate the requirement of evaluating the elliptic curve on a finite prime field by explicitly defining a finite field range and wrapping the out-of-range evaluations over the elliptic curve with the modulo operator  $\% p$ . The final form of the  $\oplus$  operation is shown in figure 2. In a real world ECC implementation, this process is carried out a large number of times (assuming  $a = b$ , i.e. take tangent at point  $a$ ) making the forward process increasingly complex but computable and the resulting inverse problem increasingly difficult to solve.

In literature, you will often find the operation  $\oplus$  equated to the generic addition  $+$ . This is not surprising since equations (2) and (3) resemble  $a + b = c$  and  $a + 0 = a$  for generic real numbers;  $0$  is the identity element of the  $+$  operation. Having said that, in practice  $\oplus$  operation is nothing like the addition operation  $+$ ; the apparent similarity lies in their algebraic representations only. For instance, equations (2) and (3) also resemble  $a \times b = c$  and  $a \times 1 = a$  for generic real numbers. In this case,  $1$  is the identity element of the  $\times$  operation. This difference in notation is a major source of confusion for the readers. In the end,  $\oplus$  should not be compared to either  $+$  or  $\times$ .

The 'large number of repetitions' is assigned as the secret private key  $K_{\text{sec}}$  and the public key  $K_{\text{pub}}$  is equivalent to calculating  $a \oplus a$  at a public value  $a = a_0$  repeated  $N$  number of times<sup>4</sup>, i.e.  $K_{\text{pub}} = ((a_0 \oplus a_0)_1 \oplus a_0)_2 \oplus a_0)_3 \dots)_N$ . The exact inverse problem – called the elliptic curve discrete logarithm problem – is equivalent to finding the number of repetitions it took to arrive at  $K_{\text{pub}}$  from a publicly known value  $a_0$ . Is this inverse problem hard? It appears that given knowledge of  $K_{\text{pub}}$  and  $a_0$ , one can easily revert the process described in figure 2 and derive key  $K_{\text{sec}}$ . This is indeed true and the inverse problem only becomes hard when the coefficients in equation (1) are restricted to the prime field  $F_p$ ; more on this in the next subsection. Some of the popular elliptic curves today are specifically `ed25519` and `secp256k1`, with the later being more recent and more robust than the former.

The name of the inverse problem for ECC, i.e. elliptic curve discrete logarithm problem, is highly misleading and results from purely notational reasons. The term 'logarithm' originates from interpreting the result of  $N$  times repetition of  $a \oplus a$  operation as  $K_{\text{pub}} = a^N$ .

It now appears that we have achieved what we set out to do: we have found a new group whose elements are now values  $y$  evaluated over the finite prime field  $F_p$  and the new group operation  $((a_0 \oplus a_0)_1 \oplus a_0)_2 \oplus a_0)_3 \dots)_N (\equiv a_0^N)$  is very difficult to invert. We note that the result of  $\oplus$  operation on the group

<sup>4</sup> $a = b$  assertion is made purely for simplicity since having  $a \neq b$  doesn't provide extra security and in fact makes forward computation unnecessarily involved.

elements results in a value which is also a member of the group<sup>5</sup>; this was not the case in RSA scheme where the result  $a \times b$  is never a member of the group.

Curious observers may note that unlike RSA where we keep both  $a$  and  $b$  secret (and reveal their product  $a \times b$ ), in ECC we additionally also reveal one of the independent variables  $a_0$ , i.e.  $a_0^N$  and  $a_0$  are both known, leaving only  $N$  to be discovered. Why is this the case? The answer lies in the fact that the solutions  $a$  and  $b$  of the product  $a \times b$  are unique while the solutions  $a$  and  $N$  of the expression  $a^N$  are not unique unless either  $a$  or  $N$  is specified. Uniqueness of the keypairs is a very important property that cannot be violated for cryptosystems; this requires  $a_0$  to be fixed and specified publicly in ECC algorithms.

## ROBUSTNESS OF ECC

The natural question to ask here is: how robust is ECC when compared to RSA and Diffie-Hellman? We remember from previous section that the inverse problem in ECC only becomes hard when the coefficients  $p$  and  $q$  in equation (1) are also restricted to a large prime field. To understand why this happens, simply note that for large prime values of  $p$ ,  $q$  and  $x$  in equation (1), all terms on the right side are products of primes and then their sums. Similar to RSA, inverting this function numerically is equivalent to prime factorising many  $a \times b$  products (and 'inverting' their sums). The fact that this process will have to be repeated a very large number of times (at least  $K_{\text{sec}}$  times) to potentially brute-force for the private key, makes the ECC algorithm several orders of magnitudes more secure than RSA or Diffie-Hellman. While this is certainly an improvement, the realisation must dawn on us that the ECC is not a new flavour of cryptography but essentially an RSA-variant on steroids, in the sense that it 'supercharges' the prime factorisation problem for large prime products by redefining the group and its native operation  $\oplus$ . The increase in complexity in ECC lies entirely in the redefined native operation  $\oplus$  which is a supercharged mix of addition  $+$  and product  $\times$  among large prime or composite numbers. Breaking ECC is therefore equivalent to breaking a very large number of RSA-like atomic prime factorisations. While the incremental complexity of ECC over RSA is sufficient at the moment, it will certainly not be sufficient in the post-quantum era; this finally brings us to the world of zero-knowledge cryptography.

Before moving forward, let's recap the process of our rediscovery of cryptography. Cryptosystems are motivated by the need to prove access to information without revealing a part or the entirety of the said information; such systems form the building blocks of countless cryptographic methods such as encryption/decryption, permissionless access, secure key sharing, digital signatures, blockchains etc. Such information-preserving proving algorithms are called 'trapdoors' since such algorithms – similar to a trapdoor – are easy to compute in one direction but difficult to invert. RSA is one of the simplest implementations of such trapdoors and it relies on the (non-)invertibility of product operation  $a \times b$  on a finite prime field; this securing property is called the prime factorisation problem. RSA algorithm is relatively safe to use at the moment but it is expected that it will be broken within the next decade. ECC is the next generation of cryptographic trapdoor algorithms which essentially supercharges the prime factorisation problem by several orders of magnitude. It does so by resampling the group elements from an elliptic curve evaluated over a finite prime field and defining a new operation  $\oplus$  on this group which

<sup>5</sup>group is referred to as 'closed under the operation  $\oplus$ '





captures the supercharging effect composed of large number of atomic prime factorisations. The supercharging increases the security of the resulting trapdoor but fails to increase it beyond the post-quantum regime due to being limited by the difficulty of prime factorisation problem.

## ZERO-KNOWLEDGE CRYPTOGRAPHY

## REFERENCES

- [1] Quisquater, JJ. et al. (1990), [How to Explain Zero-Knowledge Protocols to Your Children](#), Lecture Notes in Computer Science, Vol. 435, Springer, New York, NY
- [2] Extended Euclidean Algorithm
- [3] Quadratic Sieve Algorithm
- [4] General Number Field Sieve Algorithm
- [5] SNDL Problem
- [6] Diffie-Hellman Key Exchange
- [7] Elliptic Curve Cryptography

## METADATA

Github:   
Contracts:   
Source:   
SHA-1 Checksum:   
Date: July 6, 2023 