João Pedro Congio Martins, 176117

Prof. Ricardo Dahab

MC889 - Introduction to Cryptography

July 4, 2019

Zero-knowledge Proof

1. Introduction

In cryptography, a zero-knowledge proof, or zero-knowledge protocol, is a probabilistic-based verification method that can be used to “prove” “fact-like statements” and “statements about personal knowledge” from one party, the *prover*, to another, the *verifier*, in a way that the verifier doesn't learn **anything** else about the statement except that it is true. So, by using this method, a prover could prove that they know a value x without conveying any information apart from the fact that they know the value x. The essence of zero-knowledge proofs is that it is trivial to prove that one possesses knowledge of certain information by simply revealing it; the challenge is to prove such possession without revealing the information itself or any additional information. The goal is to convince a second party without providing any of the knowledge that would allow him, or an eavesdropper, to convince a third party.

There are essentially two variants of zero-knowledge protocol.

An *interactive zero-knowledge proof*, as the name suggests, models an interactive proof system between the verifier and the prover. The prover first issues a set of commitments, then the verifier issues challenges (that are based on certain randomness) that the prover complies with; this proves anything only as long as the verifier is assumed to issue challenges normally without any prior understanding with the prover. Moreover, this variant allows the proof of knowledge to be proven only directly to the verifier, as any third-party observer watching the interactive process or a recording of it cannot be sure the prover and the verifier haven't orchestrated the whole "experiment".

Non-interactive zero-knowledge proof, on the other hand, relies on computational assumptions (typically the assumptions of an ideal cryptographic hash function), and can be realized by using models like the *Random Oracle* model or the *Common Reference String* model to replace the verifier by a hash function or something similar, requiring no interaction between parties in the process of the proof construction. In this variant, the prover runs the protocol with a simulator (e.g.: a hash function), and then can give the transcript of the protocol to anyone and convince them that the proof is real.

In both variants, if the correct answer is given by the prover, he has a high probability of possessing what he claims to be "knowledge." The more the protocol is repeated, the more likely the prover possess that knowledge. This means that zero-knowledge proofs are not proofs in the mathematical sense of the term because there is some small probability, the *soundness error*, that a cheating prover will be able to convince the verifier of a false statement. In other words, zero-knowledge proofs are probabilistic "proofs" rather than deterministic proofs. However, there are techniques to decrease the soundness error to negligibly small values.

With respect to cryptographic protocols, generally, communicating the knowledge efficiently is not the focus, but the whole problem is making sure not too much knowledge has been communicated. The unique aspect of zero-knowledge proofs is that they allow sensitive data to be verified without revealing that data or any additional information about it, which means they are particularly effective in secure communication, authentication, and privacy. They therefore have the potential to revolutionize the way data is collected, used and transacted with.

2. Historical background

Zero-knowledge proofs were first introduced in 1989 by Shafi Goldwasser, Silvio Micali, and Charles Rackoff in their paper "The Knowledge Complexity of Interactive Proof-Systems", as a “new efficient method of communicating a proof”. This paper introduced the IP (Interactive Polynomial-time, the class of languages possessing an interactive proof system) hierarchy of interactive proof systems and conceived the concept of *knowledge complexity*, a measurement of the amount of knowledge about the proof transferred from the prover to the verifier. They also gave the first zero-knowledge proof for a concrete problem, that of deciding quadratic nonresidues mod m (this more or less means that there isn't any number x where x^2 is "equivalent" to some given number). Together with a paper by László Babai and Shlomo Moran, this landmark paper invented *interactive proof systems,* for which all five authors won the first Gödel Prize in 1993. Their primary concern centered around information leakage, meaning how much information will a verifier learn over the course of verifying that a claim is valid.

Prior to Goldwasser et al., most work in this area focused the soundness of the proof system. That is, it considered the case where a malicious prover attempts to ‘trick’ a verifier into believing a false statement. What Goldwasser, Micali and Rackoff did was to turn this problem on its head. Instead of worrying only about the Prover, they asked: what happens if you don’t trust the Verifier?

In their own words, Goldwasser, Micali, and Rackoff say:

“Of particular interest is the case where this additional knowledge is essentially 0 and we show that [it] is possible to interactively prove that a number is quadratic non residue mod m releasing 0 additional knowledge. This is surprising as no efficient algorithm for deciding quadratic residuosity mod m is known when m’s factorization is not given. Moreover, all known NP proofs for this problem exhibit the prime factorization of m. This indicates that ***adding interaction to the proving process, may decrease the amount of knowledge that must be communicated in order to prove a theorem****.”*

The quadratic nonresidue problem has both an NP and a co-NP algorithm, and so lies in the intersection of NP and co-NP. This was also true of several other problems for which zero-knowledge proofs were subsequently discovered, such as an unpublished proof system by Oded Goldreich verifying that a two-prime modulus is not a *Blum* integer.

Oded Goldreich, Silvio Micali, and Avi Wigderson took this one step further, showing that, assuming the existence of unbreakable encryption, one can create a zero-knowledge proof system for the NP-complete graph coloring problem with three colors. Since every problem in NP can be efficiently reduced to this problem, this means that, under this assumption, **all problems in NP have zero-knowledge proofs**. The reason for the assumption is that, as in the above example, their protocols require encryption. A commonly cited sufficient condition for the existence of unbreakable encryption is the existence of one-way functions, but it is conceivable that some physical means might also achieve it.

On top of this, they also showed that the graph nonisomorphism problem, the complement of the graph isomorphism problem, has a zero-knowledge proof. This problem is in co-NP, but is not currently known to be in either NP or any practical class. More generally, Russell Impagliazzo and Moti Yung as well as Ben-Or et al. would go on to show that, also assuming one-way functions or unbreakable encryption, there are zero-knowledge proofs for *all* problems in IP = PSPACE, or in other words, **anything that can be proved by an interactive proof system can be proved with zero knowledge**.

Not liking to make unnecessary assumptions, many theorists sought a way to eliminate the necessity of one way functions. One way this was done was with multi-prover interactive proof systems, which have multiple independent provers instead of only one, allowing the verifier to "cross-examine" the provers in isolation to avoid being misled. It can be shown that, without any intractability assumptions, all languages in NP have zero-knowledge proofs in such a system.

It turns out that in an Internet-like setting, where multiple protocols may be executed concurrently, building zero-knowledge proofs is more challenging. The line of research investigating concurrent zero-knowledge proofs was initiated by the work of Dwork, Naor, and Sahai. One particular development along these lines has been the development of witness-indistinguishable proof protocols. The property of witness-indistinguishability is related to that of zero-knowledge, yet witness-indistinguishable protocols do not suffer from the same problems of concurrent execution. Now, their integration into decentralized networks is pushing their application even further.

Based on all the good properties provided by the use of efficient zero-knowledge proofs in interactive proof systems and by the fact that hard problems of pure mathematics are used electronically in the practical applications, many applications for these protocols were suggested by mathematicians. These applications usually involved some form of secure communication, authentication, or privacy.

In an article for the New York Times back in 1987, the mathematicians and cryptologists behind ZK proofs already suggested uses in signing contracts, credit-card verification, password verification, and even in protecting aircraft signals.

In 2016, the Princeton Plasma Physics Laboratory and Princeton University demonstrated a novel technique based on ZK proofs that may have applicability to future nuclear disarmament talks. It would allow inspectors to confirm whether or not an object is indeed a nuclear weapon without recording, sharing or revealing the internal workings which might be secret.

Today, cryptocurrencies and blockchains are a hot field where ZK protocols are being used for anonymity in transactions. Platforms that use some form of zero-knowledge proofs include Zcash, Monero, PIVX, and Zerocoin. Importantly, these cryptocurrencies use zero-knowledge proofs to obfuscate the details of transactions on public blockchain networks. These details include the sender, recipient, and the amount transferred.

3. Mathematical background

Following are some basic definitions and properties of cyclic groups, as well as an introduction to the Discrete Logarithm Problem. These are of great importance to the protocols that will be explored later in the text, serving as a building block for them to actually work.

3.1. Definition: Order of an element

The order of an element *a* of a group is the smallest positive integer *m* such that $a^m = e$, where *e* denotes the identity element of the group, and a^m denotes the product of *m* copies of *a*.

3.2. Definition: Cyclic Group

A group *G* which contains an element α with maximum order $ord(\alpha) = |G|$ is said to be cyclic. Elements with maximum order are called primitive elements or *generators*. An element α of a group *G* with maximum order is called a generator since every element *a* of *G* can be written as a power $\alpha^i = a$ of this element for some *i*, i.e., α generates the entire group.

Theorem: For every prime *p*, $(Z_p^*, \cdot)$ is an abelian finite cyclic group. In other words, the multiplicative group of every prime field is cyclic.

3.3. Definition: Discrete Logarithm Problem (DLP) in Z_p^*

Given is the finite cyclic group Z_p^* of order $p - 1$ and a primitive element $\alpha \in Z_p^*$ and another element $\beta \in Z_p^*$. The DLP is the problem of determining the integer $1 \le x \le p - 1$ such that: $\alpha^x \equiv \beta \ mod \ p$. The integer *x* is called the discrete logarithm of *β* to the base *α*, and we can formally write: $x = log_\alpha \beta \ mod \, p$. Computing discrete logarithms modulo a prime is a very hard problem.

4. Zero-knowledge proofs basics

A zero-knowledge proof must satisfy three properties:

1. **completeness**: if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover;
2. **soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability, namely, the *soundness error*.
3. **zero-knowledge**: if the statement is true, no verifier learns anything other than the fact that the statement is true. In other words, just knowing the statement (not the secret) is sufficient to imagine a scenario showing that the prover knows the secret. This is formalized by showing that every verifier has some simulator that, given only the statement to be proved (and no access to the prover), can produce a transcript that "looks like" an interaction between the honest prover and the verifier in question.

Let us introduce the basics of zero-knowledge proofs with an example protocol for graph three-colouring, originally devised by Goldreich, Micali and Wigderson (GMW). This interactive protocol allows a prover to prove, in zero-knowledge, that a graph supports a three-coloring. We assume an honest prover *P* and an honest verifier *V* under a commitment scheme, and a graph *G* with set of edges *E*. The protocol is operated as follows:

- The honest prover *P* has access to a 3-coloring σ for *G*. He randomly chooses a permutation $\pi \leftarrow perm\{0,\ 1,\ 2\}$ of colors and commits to $\pi(\sigma)$;
- The honest verifier *V* then chooses a random edge *(i, j)* from *E* and sends this choice to *P*;
- *P* then reveals to *V* the permuted colours of vertices *i* and *j*, namely $\pi(\sigma_i)$ and $\pi(\sigma_j)$ and the randomness r_i and r_j he used to commit these colours;
- *V* checks if the original commitments match the revealed values, and that the two colours $\pi(\sigma_i)$ and $\pi(\sigma_j)$ are different. *V* accepts if both these conditions hold.

At first, this procedure doesn't seem very helpful in proving a three coloring for G. After one run of this protocol, P could succeed in cheating V with probability up to $(|E| - 1)/|E|$ (which for a 1,000 edge graph works out to 99.9% of the time). In other words, a cheating prover must commit to a colouring that has at least 1 edge whose two endpoints are coloured with the same colour and, with probability $1/|E|$, the verifier catches him whenever he happens to request the colours for this edge, due to the binding property of the commitment scheme. The key idea, though, is to repeat the protocol multiple times to reduce the soundness error to a negligible small value. By performing $n|E|$ sequential repetitions of the protocol, we can get the soundness error down to $(1 - \frac{1}{|E|})^{n|E|} \approx e^{-n}$, which is negligible. So, the soundness property is verified.

Completeness proof is straightforward: The honest prover can answer all of the verifier's queries correctly such that the honest verifier will be convinced as long as the commitment scheme is binding.

To prove the zero-knowledge property, we need to show that every verifier has some simulator that, given only the statement to be proved, can produce a transcript that "looks like" an interaction between the honest prover and the verifier in question. For this, we will use a special kind of prover, i.e., a simulator, operating as follows:

- Given a graph $G = (V, E)$, simulator S chooses an edge (i, j) at random from E. He colours i and j different colours, and colours the rest of the vertices the same colour, say colour 0. He then permutes the colours randomly, and commits to this permuted colouring, just as the honest prover does.
- He then simulates the verifier V on these committed colourings, and receives (i_0, j_0), the first message V sends.
- If $(i, j) = (i_0, j_0)$, then the simulator can honestly answer the query, and thus simulates the remainder of the protocol, and outputs the transcript.
- If $(i, j) \neq (i_0, j_0)$, then the simulator restarts, from the first step, with a fresh (i, j).

Notice that the choice of (i_0, j_0) makes must be (i, j) with probability very close to $1/|E|$, since the only information it is given are the commitments, which intuitively cannot bias its decision because of the hiding property. In other words, a V that outputs (i, j) with probability very different from $1/|E|$ can be used to break the multi-message hiding property of the commitment scheme. It follows that the simulator succeeds with probability close to $1/|E|$, and thus has expected number of iterations close to $|E|$, which means it runs in expected poly-time. This means that a verifier wouldn't be able to tell the difference in runtime between the simulator and an honest prover. We also need to show that the output of the simulator is indistinguishable from the output of an execution of the protocol with an honest prover. The colours are certainly correctly distributed, since they are just two random different colours, and the edge requests are approximately correctly distributed as well. This proves the last property (zero-knowledge) and thus the protocol is a valid zero-knowledge protocol.

Note that the real significance of the GMW result is theoretical. Since graph three coloring is known to be in the complexity class NP-complete, the GMW protocol can be used to prove any statement in the class NP. This means that, if there exists any decision problem (that is, a problem with a yes/no answer) whose witness (solution) can be verified in polynomial time, then we can prove that said solution exists by (1) translating the problem into an instance of the graph three-coloring problem, and (2) running the GMW protocol. However, all this process is very expensive in practice.

A more useful proof of knowledge exists. It can serve as a useful building block for many cryptographic protocols to ensure that participants follow the protocol specification honestly. In fact, it's the basis of many modern signature schemes today. We now introduce the Schnorr identification protocol, invented by Claus-Peter Schnorr in the 1980s. What it achieves is the proof of knowledge of a *discrete logarithm*, a very useful concept in cryptography.

5. The Schnorr identification protocol

Just like the previous graph three-colouring protocol example, the Schnorr identification protocol is part of a set of protocols called Sigma protocols which follow a three-round interactive proof structure between two parties *A* and *B: A* commits to a value, *B* challenges that value, and *A* responds the challenge. The Greek letter "Σ" visualizes the three-way flow of the protocol.

Imagine that Alice has published her public key to the world, and later on wants to prove that she knows the secret key corresponding to that public key. This is the exact problem that we encounter in real-world protocols such as public-key SSH, so it turns out to be well-motivated.

First, we will explore an interactive version of the Schnorr identification protocol, where its zero-knowledge properties are more evident.

Schnorr began with the assumption that the public key would be of a very specific format. Specifically, let *p* be a prime number, and let *g* be a generator of a cyclic group of prime-order *q*. To generate a keypair, Alice would first pick a random integer *a* between 1 and *q*, and then compute the keypair as:

$$PK_A = g^a \ mod\, p,\ SK_A = a\,,$$

where PK_A is Alice's public key and SK_A is Alice's private key.

(Note that this is the same type of key used for Diffie-Hellman and the DSA signing algorithm, both based on the Discrete Logarithm Problem).

Now, assume Alice wants to prove knowledge of her private key to Bob. They conduct the following interactive protocol:

- Alice picks random *k* in range $\{1,\ ...,\ q\}$ and sends $h = g^k \ mod\, p$ to Bob;
- Bob picks random *c* in range $\{1,\ ...,\ q\}$ and sends *c* to Alice;
- Alice sends $s = ac + k \ mod\ q$ to Bob;
- Finally, Bob checks if $g \equiv PK_A{}^c \cdot h \ mod\, p$.

To show that this is indeed a zero-knowledge proof, we now verify the three properties: completeness, soundness and zero-knowledgeness.

Completeness is straightforward. By following the protocol, we get:

$g^{s} \equiv PK_A{}^{c} \cdot h \; mod \; p$

$g^{ac+k} \equiv (g^{a})^{c} \cdot g^{k} \; mod \; p$

$g^{ac+k} \equiv g^{ac+k} \; mod \; p$

So, after completion, Bob should be satisfied with the proof.

To prove soundness, we must show that if Alice successfully convinces Bob, then she must know the secret key *a*. For that, we assume a special type of verifier that, upon interacting with a prover, and if the prover succeeds in completing the proof, is able to extract the prover's secret. We call this verifier a *knowledge extractor* and, just like the simulator from the graph example, it should be able to restart the prover to a specific step in the protocol. To prove soundness for a proof of knowledge, it is sufficient to show that an extractor exists for every possible prover. Suppose then Alice follows the protocol with an extractor *E* that is able to rewind Alice to a specific step in the protocol:

- Alice picks random *k* in range $\{1, \; ..., \; q\}$ and sends $h = g^{k} \; mod \; p$ to *E*;
- E picks a random c_1 in range $\{1, \; ..., \; q\}$ and sends c_1 to Alice;
- Alice sends $s_1 = a(c_1) + k \; mod \; q$ to *E*;
- *E* rewinds Alice to step 2;
- *E* picks a different random c_2 in range $\{1, \; ..., \; q\}$ and sends c_2 to Alice;
- Alice then sends $s_2 = a(c_2) + k \; mod \; q$ to *E*.

Now, the extractor can solve the following equation ro recover Alice's secret key:

$(s_1 - s_2)/(c_1 - c_2) \; mod \; q$

$\equiv ((ac_1 + k) - (ac_2 + k))/(c_1 - c_2) \; mod \; q$

$\equiv a(c_1 - c_2)/(c_1 - c_2) \ mod \ q$

$\equiv a$

The key observation here is that by rewinding Alice's execution, the extractor can 'trick' Alice into making two different proof transcripts using the same *k*. This shouldn't normally happen in a real protocol run, where Alice specifically picks a new *k* for each execution of the protocol.

To prove zero-knowledgeness, the last property, we require a simulator that can interact with any possible verifier and produce a 'simulated' transcript of the proof, even if the simulator doesn't know the secret it's proving it knows.

For this, we need to assume an honest verifier, that is, a verifier that will pick its challenge "*c*" using only its random number generator, and will not choose this value based on any input the prover provides it. With this assumption, the simulator can be constructed.

Assume a simulator *S* trying to prove knowledge of a secret *a* for some public key $g^a \ mod \, p$, but it doesn't actually know the value *a*. *S* assumes that the verifier will choose some value *c* as its challenge, and moreover, it knows that the honest verifier will choose the value *c* only based on its random number generator, and not based on any inputs the prover has provided. The interaction is as follows:

- *S* outputs some initial g^{k_1} as the prover's first message, and finds out what challenge *c* the verifier chooses;
- *S* restarts the verifier, and picks a random integer *z* in the range $\{0, \ ..., \ q - 1\}$;
- *S* computes $g^{k_2} = g^z \cdot g^{a(-c)}$ and output g^{k_2} as the Prover's new initial message.
- When the verifier challenges on *c* again, *S* outputs *z*.

Notice that the transcript $g^k, \ c, \ z$ will verify correctly as a perfectly valid, well-distributed proof of knowledge of the value *a*. The verifier will accept this output as a valid proof of knowledge of *a*, even though *S* does not know *a*. Since the statistical distribution of this protocol is identical to the real protocol, the zero-knowledgeness property is verified.

This concludes the proof that the Schnorr identification protocol is a valid ZK protocol. Although this protocol is useful, it requires an interactive verifier for a prover to use the protocol. The interaction itself is also quite expensive. We now move on to a more practical version of the Schnorr protocol, which requires no such interaction.

5.1. Non-interactive Schnorr protocol

To transform an interactive protocol to a non-interactive protocol we can use the Fiat-Shamir transformation, a technique for minimizing interaction in interactive protocols developed by Fiat and Shamir in the 1980s. What is needed for this transformation is a special hash function. More precisely, we need a hash function that is a *random oracle*: a mathematical function chosen uniformly at random, that is, a function mapping each possible query to a (fixed) random response from its output domain.

The biggest difference between the interactive version and the non-interactive version is that, in the non-interactive version, the challenge *c* will come from the hash function, and not from a verifier.

Using the Fiat-Shamir transformation and assuming a random oracle as a hash function *H()*, the new Schnorr protocol for proving knowledge of *a* with respect to a public key g^a looks like this:

- The prover picks g^k (just as in the interactive protocol).
- Now, the prover computes the challenge as $c = H(g^k \| M)$ using the hash function *H*, where *M* is an (optional) and arbitrary message string.
- Compute $ac + k \ mod \ q$ (just as in the interactive protocol).

As we can see, this protocol is much simpler, but it needs the assumption of a random oracle.

When realized like this, the protocol isn't just a proof of knowledge, it's also a signature scheme. That is, if a valid message string *M* is used, the signature on *M* is obtained, which can only be produced by someone who knows the secret key *a*. The resulting protocol is called the Schnorr signature scheme, and it's the basis of real-world protocols like EdDSA.

To close the discussion on zero-knowledge proofs we talk about one of it's newest application *zk-SNARKs*, a technology used for privacy in *blockchains*.

6. The zK-SNARKs technology

Most crypto-asset holders don't want their holdings and transaction history to be completely public. Among the various cryptographic techniques aiming to provide privacy to blockchains, the zk-SNARK proof is a noteworthy example.

The acronym zk-SNARK stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge":

- **Zero-knowledge:** has the zero-knowledgeness property of a ZK proof;
- **Succinct:** the sizes of the messages are tiny in comparison to the length of the actual computation, so a "succinct" proof can be verified within a few milliseconds, with a proof length of only a few hundred bytes even for statements about programs that are very large;
- **Non-interactive:** there is no or only little interaction. For zkSNARKs, there is usually a setup phase and after that a single message from the prover to the verifier. Furthermore, SNARKs often have the so-called "public verifier" property meaning that anyone can verify without interacting anew, which is important for blockchains;
- **Arguments:** the verifier is only protected against computationally limited provers. Provers with enough computational power can create proofs/arguments about wrong statements (Note that with enough computational power, any public-key encryption can be broken). This is also called "computational soundness", as opposed to "perfect soundness";

- **of Knowledge:** it is not possible for the prover to construct a proof/argument without knowing a certain so-called witness (for example the address she wants to spend from).

In order to have zero-knowledge privacy for users in a network for cryptocurrencies transactions, the function determining the validity of a transaction according to the network's consensus rules must return the answer of whether the transaction is valid or not, without revealing any of the information it performed the calculations on. This is done by encoding some of the network's consensus rules in zk-SNARKs. At a high level, zk-SNARKs work by first turning what a party wants to prove into an equivalent form about knowing a solution to some algebraic equations.

The cryptocurrency *Zcash* is the first widespread application of zk-SNARKs. The strong privacy guarantee of Zcash is derived from the fact that shielded transactions in Zcash can be fully encrypted on the blockchain, yet still be verified as valid under the network's consensus rules by using zk-SNARK proofs.

7. Closing comments

What Goldwasser, Micali and Rackoff proposed back in 1989 was a hope for revolutionizing the concept of proofs and how they can be constructed. If fully realized, it would allow us to prove statements about personal knowledge in a way that is as powerful as it is counterintuitive. The own definition of zero-knowledge was something that was appealing to cryptography from the start, due to its strong privacy and secrecy promises.

Zero-knowledge proofs will continue to be applied wherever they are useful as they continue to develop. The underlying technology may be extremely complex, but their potential for privacy, authenticity, and security cannot be overstated. Today, the use of zero-knowledge proofs in cryptocurrencies is pushing the innovation of the technology even further.

8. References

[1] Shafi, Goldwasser et al. *The Knowledge Complexity of Interactive Proof-Systems*, 1989.

[2] Gleick, James. “A NEW APPROACH TO PROTECTING SECRETS IS DISCOVERED”. *The New York Times*, 1987.

[3] Green, Matthew. “Zero Knowledge Proofs: An illustrated primer”, *A Few Thoughts on Cryptographic Engineering,* 27 November 2014.
https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/

[4] Wikipedia. “Zero-knowledge proof”. Accessed 4 July 2019.
https://en.wikipedia.org/wiki/Zero-knowledge_proof

[5] Wikipedia. “Proof of knowledge”. Accessed 4 July 2019.
https://en.wikipedia.org/wiki/Proof_of_knowledge

[6] Wikipedia. “Interactive proof system”. Accessed 4 July 2019.
https://en.wikipedia.org/wiki/Interactive_proof_system

[7] Wikipedia. “Random oracle”. Accessed 4 July 2019.
https://en.wikipedia.org/wiki/Random_oracle

[8] Pass, Rafael. “Cryptography Computer Science 6830” course at Cornell University, Fall 2011.

http://www.cs.cornell.edu/courses/cs6830/2011fa/

[9] Damgard, Ivan. *On Σ-protocols*, 2010. http://www.cs.au.dk/~ivan/Sigma.pdf

[10] Reitwiessner, Christian. “zkSNARKs in a nutshell”. *Ethereum Blog*, 5 December 2016

https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/

[11] Zcash. “What are zk-SNARKs?”. Accessed 4 July 2019.

https://z.cash/technology/zksnarks/