Zero Knowledge Proof: What Can Go Wrong?

***Abstract*—The advancement of cryptography has allowed the concept of Zero Knowledge Proof (ZKP) to nurture. Despite its astonishing concept, it requires a good amount of knowledge to understand its concept. Without intuitive explanation, it is hard to spark interest in this specific field. In this paper, we introduce a narrative as a subtle introduction into ZKP.**

Without proper understanding, it is also easy to create mistakes during the implementation of ZKP protocols, which can lead to security concerns. One recent security phenomenon in ZKP is the Frozen Heart Vulnerability. In this paper, we also discuss this vulnerability deeper and try to understand possible causes and fixes for this vulnerability.

***Index Terms*—security, cryptography, zero-knowledge**

I. Introduction

One aspect of security that has been discussed more often lately is privacy. Awareness of thge privacy of individuals is higher than ever before. This also affects the development of technology. Respecting privacy has become a requirement in many cases, which pushed the development of new systems that protect privacy while not compromising other security requirements.

Cryptography is one key enabler for such technology. This also includes the concept of ZKP. The recognition of ZKP has been on the rise since cryptocurrency has its stage within society. However, the application of ZKP is not limited to cryptocurrency.

Recently, we were faced with the SARS-CoV-2 pandemic. The transmissibility of this virus combined with its characteristics make this virus widely spread all over the world. This calls for a prevention mechanism using contact tracing. But such methods raise questions about privacy of users, because they may require knowing the connection between people. To circumvent such problem, Google and Apple applied technology based on ZKP concept to allow authorities to trace contact of an infected person [1].

The ZKP protocol is also useful in politics. With the rise in the number of eligible voters, the manual vote-counting method became far more costly to be done. The number of human and financial resources required to conduct voting does not always scale properly. Moreover, the more actors involved in the system, the higher the risk of cheater might exist.

To answer such problem, technologies like electronic voting can be implemented. However, this method creates doubt about the anonymity of the vote. To be able to implement a fair and anonymized voting system, the ZKP protocol has been used by Hao *et al.* [2] and McCorry *et al.* [3] in their works. Their works allow verifying whether a person has cast a vote or not. This way, it can prevent a person to vote more than once. They also prevent a cheating actor to map a vote to the voter, respecting the anonymity aspect of voting.

From the aforementioned examples, we can see that the ZKP has a vast range of applications. This is possible because ZKP is more than a concrete protocol, but rather a concept based on mathematical notion. It unlocks the possibility of a concept that is hard to be perceived at first glance.

Due to its mathematical and abstract nature, it is not easy to grasp the concept of ZKP without good mathematical background. This makes it harder to wake people's interest and decelerates the adoption of ZKP in everyday use cases. It also requires a strong understanding of the concept to be able to properly implement a secure protocol with ZKP concept. Unfortunately, we found that there is too few literatures with intuitive explanation about the concept of ZKP.

A misunderstanding of the concept can lead to vulnerabilities in the end implementation. One discovery of such vulnerability was published by Miller in his blog post. In his blog, he mentioned possible exploit of ZKP libraries that has security flaws in its core, which he mentioned as the Frozen Heart vulnerability. It underscores the possible threat if the concept of ZKP is not well understood during implementation.

Despite its severity, the Frozen Heart vulnerability lacks of proper acknowledgements in the scientific community. There is also insufficient attention for this vulnerability in scientific literature. To address these problems, we published this paper as a brief introduction to the ZKP concept. We will discuss an intuitive and narrative example of ZKP schema that can be taught and performed for didactic purposes. We also explain the background that coined the concept of ZKP and several interesting protocols which build upon ZKP concept.

This papers will also answer following research questions:

1) What is Zero Knowledge Proof?
2) What is the Frozen Heart vulnerability?
3) What are possible fixes of the Frozen Heart vulnerability?
4) What can be learned from the Frozen Heart vulnerability?

This paper is constructed as follows: We list and discuss preceding works related to this topic and explain the relevance of our paper in Section II. In Section III we will discuss the concept of ZKP. Section V discusses the Frozen Heart Vulnerability, which includes its causes, impacts, and possible fixes of this vulnerability. Lastly, we discuss important key takeaways from the Frozen Heart Vulnerability in Section VI.

A. Notations and Symbols

We use several symbols and mathematical formulae throughout the paper. Hence, we introduce them here to achieve consistency and clarity. In this paper, we will use several actors for simulating the ZKP protocol. Unless specified otherwise, we use $\mathcal{P}$ as the prover and $\mathcal{V}$ as the verifier. We also introduce $\mathcal{T}$ as a trust center, which acts mainly to support the protocol execution between $\mathcal{P}$ and $\mathcal{V}$.

For variables in the protocol, we use r mainly for random values unless specified otherwise. We also use the symbol C for commitment in the protocol meanwhile P denotes the proof. For this paper, we also define several mathematical symbols. Unless specified otherwise, we use $[n]$ to define a set of number from 0 until n, formally written $[n] = \{0, 1, \ldots, n\}$. However, we will more often use $[n]^+ = [n] \backslash \{0\}$.

In the later part of this paper, we use vectors similarly to [4]. We write vector in bold, e.g., $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$. A multiplication between scalar x and vector $\mathbf{a}$ is defined by multiplying all entries of $\mathbf{a}$ with x. The operator $\circ$ defines the Hadamard product or element-wise multiplication of two vectors, i.e., $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_n \cdot b_n)$. We use $\langle \mathbf{a}, \mathbf{b} \rangle$ to define inner-product of two vectors $\mathbf{a}$ and $\mathbf{b}$, formally described $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i$. For integers x and n, we define $\mathbf{x}^n = (x^0, x^1, \ldots, x^n)$. For a cyclic group $\mathbb{G}_p$ of prime order p and $g = (g_1, \ldots, g_n) \in \mathbb{G}_p^n$, we also define $\mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{n} g_i^{a_i} \in \mathbb{G}_p^n$.

Lastly, we define the polynomial $a(X)$ as polynomial of variable X, which can be written as $\sum_{i=1}^{n} \mathbf{a}_i \cdot X^i$. The inner-product of two polynomials $\mathbf{a}$ and $\mathbf{b}$ is written $\langle a(X), b(X) \rangle$ where

$$\langle a(X), b(X) \rangle = \sum_{i=0}^{n} \sum_{j=0}^{i} \langle a_i, b_j \rangle \cdot X^{i+j}$$

II. Related Works

Flaws related to the implementation of ZKP are not new phenomena. There are several papers mentioning vulnerabilities of ZKP protocols due to flawed implementation. In this section, we discuss past vulnerabilities that were caused by flaws in the implementation of ZKP protocols.

In 2012, Bernhard *et al.* mentioned in their work [5] the vulnerability of weak Fiat-Shamir transformation for ZKP. They specifically mentioned a vulnerability in the implementation of Helios cryptographic voting protocol. The problem happens because a property is missing from the proof, which allows an attacker to create a fake proof and the validator will still accept it. The vulnerability in this paper is similar to the vulnerability discussed by Bernhard. However, we will also discuss implementation flaws found in different libraries.

In April 2022, Miller published a series of blog posts [6]–[9]. In his posts, he explained his findings of insecure implementations of ZKP protocol. He also mentioned the term "Frozen Heart" in one of his posts [6] which will also be explained in the later part of this paper. Despite having a very good explanation of the vulnerability, understanding the posts requires various knowledge about the ZKPs which is not provided in the posts. We will introduce the reader with the explanation of ZKP protocols to provide enough knowledge base to make understanding the vulnerability easier.

Attempts in creating a better explanation of ZKP protocols also have been done. Petkus explained in his work [10] the background and concept of zk-SNARK, one variant of ZKP protocol. He utilized many graphical explanations to help the reader understand the concept of zk-SNARK. In contrast to his work, we will discuss the broader concept of ZKPs in this paper.

Amit Sahai, Ph.D., also discussed several examples of ZKP explanations in a *YouTube* video [11]. His explanation includes a version similar to our narration. In contrast to his explanation, ours is more elaborated and formalized, such that it suits to be explained more formally.

III. Zero Knowledge Proof

The ZKP concept is one main discussion of this paper. It has many concrete protocols and proof methods. Mainly, the ZKP is divided into two main types, interactive proof and non-interactive proof. In this section, we go through the explanation of both interactive and non-interactive versions of ZKPs.

A. Interactive Zero Knowledge Proof

There are various explanations that explain the concept of interactive ZKP. One way is by using the famous Alibaba Cave [12]. Despite being well-known as a way to explain the concept of ZKP, this method is hard to demonstrate in our opinion. Thus, we introduce a new way to explain the concept of ZKP that is easier to be demonstrated. To do this, we use a simplified version of the puzzle "Where's Wally".

In this paper, we define the simplified version "Where's Wally" puzzle as a problem where there exist at least 2^n distinct persons in a picture. In the picture, there is exactly one person named Wally that has distinctive characteristics known to everyone. We also define that it is computationally easy to distinguish Wally and not Wally in our picture. The solution of our "Where's Wally" puzzle is a coordinate s that describes the position of Wally in the picture.

In order to find Wally without knowing s, we have to iterate through 2^n person and check whether the characteristic matches the characteristic of Wally. As a consequence, the complexity grows at $\mathcal{O}(2^n)$. However, it is very easy to verify whether the person at a certain coordinate in the picture is indeed Wally or not.

For the sake of consistency, we also introduce additional actors for our narration. In our scenario, Peggy (the prover) wants to show Victor (the verifier) that she knows the Wally's position s in a "Where's Wally" puzzle x. Yet, Peggy does not want to tell anyone (including Victor) about the position s.

To prove Peggy's claim, Peggy sends Victor a copy of her puzzle x. Later Victor takes a sheet of paper and creates 2^n markings. This paper must be big and thick enough, such that Peggy is able to position the picture behind the paper with Wally on the marking without having any part of the picture visible to the other side of the paper. This is important because Peggy does not want to give any clues about Wally's position to Victor.

Now Victor needs to create a hole in any marking. He has to create a hole in a way such that Peggy cannot predict its location. To ensure this, Victor uses a coin. He tosses a coin n times and interprets the result as a binary representation of an integer r. He then makes a hole in the marking that corresponds to r. It is important that Peggy cannot guess with

high certainty the location of the hole. Else she has the chance to forge proof.

Afterward, Victor sends the holed paper to Peggy as a challenge and asks her to put the picture behind the paper, such that he can find Wally from the hole. Peggy provides the stack of the picture and paper as proof to Victor. If Victor can find Wally in the hole, he might accept Peggy's proof is valid. If Wally is not to be found in the hole, he will reject Peggy's claim.

In the end, Peggy can provide proof to Victor that she knows Wally's position s in the picture without telling him the exact value of s. This is possible because Peggy masks the real answer within the proof P in such way that P can only be generated if she knows s but it is computationally infeasible to calculate s from P.

Although it seems secure at first glance, our protocol is highly trustable yet. Let us introduce a new person Mallory to our example. Mallory does not know the Wally's position s in the picture but wants to convince Victor otherwise. To convince Victor, Mallory can just randomly put the picture x and show Victor the proof. In fact, there is a very small probability $\frac{1}{2^n}$ where Mallory successfully places the picture such that Wally can be found within the hole.

To tackle this, Victor is not directly convinced by any prover the first time it successfully provides correct proof. Instead, he asked the prover again to perform a similar task but with different paper with a different hole position. They repeat this ask-and-answer process t times until Victor is certain that the prover knows the position of Wally in the picture. This action makes the proof system to be interactive.

Goldwasser *et al.* formalized in their work [13] such proofing scheme as a Zero Knowledge Proof (ZKP). Based on their work, a proof system $(\mathcal{P}, \mathcal{V})$ is considered as a ZKP if there exists a prover $\mathcal{P}$ which can solve an NP problem within finite time and a verifier $\mathcal{V}$ which can verify an answer to the problem also within finite time. Additionally, the proof system also has to fulfill the following properties for n the number of coin throws (or the length of random bits) and for a security parameter $k > 0$:

- The completeness property: $\mathcal{V}$ shall correctly accept a valid proof from $\mathcal{P}$ with probability at least $1 - \left(\frac{1}{n}\right)^k$
- The soundness property: $\mathcal{V}$ is only allowed to falsely accept an invalid proof from $\mathcal{P}$ with probability at most $\left(\frac{1}{n}\right)^k$
- The zero-knowledge property: $\mathcal{P}$ and $\mathcal{V}$ shall not communicate anything about the secret s.

Informally described, the completeness property requires that the proof scheme ensures the verifier accepts valid proof within a very small margin of error. The soundness property in contrast only allows with a very low probability that the verifier accepts a wrong proof as a valid one.

Unfortunately, the definition of zero-knowledge property by Goldwasser *et al.* does not prevent the verifier $\mathcal{V}$ to learn anything about $\mathcal{P}$. This is because the definition of knowledge complexity in their work only limits the communication between $\mathcal{P}$ and $\mathcal{V}$ during one ZKP phase.

Goldreich and Oren claimed in their work [14] that Goldwasser's definition can create problems when multiple ZKP protocols are used sequentially. They expect that sequential use of ZKP protocols should result in a ZKP protocol. However, it is possible under zero-knowledge definition from Goldwasser *et al.* that $\mathcal{V}$ learns clues about s from $\mathcal{P}$ during the chaining of ZKPs.

Considering the vulnerability that might arise, Goldreich and Oren propose a better definition of zero-knowledge properties. Their new definition requires that every information that $\mathcal{V}$ can compute from the communication with $\mathcal{P}$, it can compute by itself without using ZKP protocol. This means that there is no way for $\mathcal{V}$ to be able to compute or reconstruct s from communication with $\mathcal{P}$.

From our narration and definition of ZKP by Goldwasser *et al.* , we can see that the security of a ZKP protocol is assumed by the hardness of the problem that needs to be solved and the randomness of the verifier's challenge. The important property of the proof is that a verifier shall be able to verify a solution (or a proof) in a computationally feasible manner. We know that such problems exist within the NP complexity space.

In addition to such property, it is also interesting to prevent a malicious actor to be able to generate a valid proof y without knowing the secret value s. Hence, we can choose a problem from the NP space where it is not in the $\mathcal{P}$ space, with the assumption that such problem exists.

The NP problem used by Goldwasser *et al.* in their work is the quadratic non-residuosity problem. To properly discuss this problem, it is important to know the definition of quadratic residue, Legendre symbol, and Jacobi symbol.

Definition 1: An integer $x \in \mathbb{Z}_m^*$ is a quadratic residue modulo m (denoted $x \ \rho \ m$) if there exists a y where $y^2 \equiv x \mod m$ and x is relatively prime to m (this means $\gcd(x, m) = 1$), otherwise x is quadratic non-residue of m (denoted $x \not\rho m$) [15].

Definition 2: Let p a prime number and integer number x, a Legendre symbol is defined as follows [16]:

$$\left(\frac{x}{p}\right) = \begin{cases} 0 & x \equiv 0 \mod p \\ +1 & x \ \rho \ p \text{ and } a \not\equiv 0 \mod p \\ -1 & x \not\rho \ p \text{ and } a \not\equiv 0 \mod p \end{cases}$$

Definition 3: Let m an odd positive integer and $m = \prod_{i=1}^{c} p_i^{a_i}$ for prime numbers $p_1, \ldots, p_c$ and integers $a_i, \ldots, a_c$, a Jacobi symbol of x modulo m is defined as follows [17]:

$$\left(\frac{x}{m}\right) = \prod_{i=1}^{c} \left(\frac{x}{p_i}\right)^{a_i} \tag{1}$$

The right side of the equation in the eq. (1) is not a multiplication of integers' division, but rather a multiplication of Legendre symbols. Intuitively explained, the Jacobi symbol of x to m is the result of the multiplication of all Legendre

symbols of x with each prime factor of m. After we introduced the definitions used in the Quadratic Non-Residuosity Problem, we can properly define it.

Definition 4: For Quadratic Non-Residuosity Problem of number $y \in \mathbb{Z}_m^*$ and $m \in \mathbb{N}$ is defined as follows:

$$Q_m(y) = \begin{cases} 0 & y \ \rho \ m \\ 1 & y \ \not\rho \ m \end{cases}$$

In practice, m is a product of several prime numbers $p_1, \ldots, p_n$. If the factorization of m is unknown, it is computationally hard to determine $Q_m(y)$ for a given integer y where the Jacobi symbol $\left(\frac{y}{m}\right) = +1$ [13].

In their paper, Goldwasser *et al.* also defined a concrete schema of a ZKP protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$. The schema assumes that both $\mathcal{P}$ and $\mathcal{V}$ knows two numbers y and m, such that the Jacobi symbol $\left(\frac{y}{m}\right) = +1$. The schema specifies a method for $\mathcal{P}$ to show that she knows the factorization of m.

To check this claim, $\mathcal{V}$ sends a set of challenge numbers $x = (x_1, \ldots, x_n)$ where each $x_i \in x$ can be either $Q_m(x_i) = 1$ or $Q_m(x_i) = 0$ in random order (hence $\mathcal{V}$ knows the correct answer for all $Q_m(x_i)$). If $\mathcal{P}$ knows the correct factorization of m, it can compute the correct $Q_m(x_i)$ easily, else it cannot. $\mathcal{P}$ responds with the mapping of x_i to $Q_m(x_i)$. $\mathcal{V}$ will directly reject the claim if any $Q_m(x_i)$ is wrong, else it may continue with the next iteration or accepts the claim.

We can see, that the problem in the original work about ZKP is also tailored such that it is computationally hard to solve without any information or clues about the solution, but verifying the solution is computationally easy. Thus, it is important to choose a problem with such characteristics when designing a proof mechanism for a ZKP protocol.

After discussing required concept of ZKP and relevant interactive ZKP protocols, we will discuss one of many concepts that allow more efficient implementations of ZKP protocols in section IV, the Fiat-Shamir Heuristic.

B. Non-Interactive Zero Knowledge Proof

In section III-A, we introduced the concept of interactive ZKP. Despite its usefulness, interactive ZKP has several actual shortcomings in its real-world implementation. One key disadvantage of interactive ZKP is the requirement that both the prover and verifier have to communicate at the same time. This means that any communication problem during the proof protocol can prevent the protocol to terminates successfully.

Another problem with interactive ZKP is that the proof cannot be transferred, which means a prover can only convince one verifier at a time. This creates scalability issues when there are a high number of differences between prover and verifier.

Non-Interactive ZKP addresses those problems by allowing the proof to be produced without the requirement to interact with the verifier. There are several protocols which mostly used to implement non-interactive ZKP, such as zk-SNARK and zk-STARK.

Apart from zk-SNARK and zk-STARK protocols, it is also possible to implement a non-interactive ZKP protocol from interactive ZKP protocol using a technique called Fiat-Shamir Heuristic. In the next section, we will discuss this technique further.

IV. Fiat-Shamir Heuristic

In 2000, Shamir introduced in his work [18] an identity-based scheme that allows any pair of a network to perform secure communication and message signing by exchanging neither the private nor public key of each member. This work was then improved by Fiat and Shamir in their work [19]. There are two interesting schemes that contribute to the existence of non-interactive ZKP, which will be discussed further in this subsection.

The first scheme mentioned in the [19] is the interactive identification scheme. This scheme allows the identification of an entity that respects the properties of a ZKP. Similarly to the scheme mentioned by Goldwasser *et al.* [13], the security in this scheme lies on top of the complexity to perform factorization of a particular number. The identification scheme specified by Fiat and Shamir allows a prover $\mathcal{P}$ to show that she is indeed $\mathcal{P}$ to a verifier $\mathcal{V}$, but $\mathcal{V}$ cannot convince the others that he is $\mathcal{P}$.

The interactive identification scheme requires a trust center $\mathcal{T}$ which is responsible for creating a smart card for the prover $\mathcal{P}$. For performing the operation, $\mathcal{T}$ publishes a certain number m where m is a multiplication of two secret prime numbers p and q. $\mathcal{T}$ also publishes a cryptographic hash function h which maps a random string to number in $[0, m)$. In particular, distinguishing the result of the hash function of h with random bits vector shall be computationally infeasible.

In order for $\mathcal{P}$ to be able to identify itself to $\mathcal{V}$, the trust center $\mathcal{T}$ must create a smart card for string I that contains information about $\mathcal{P}$ (i.e., name, address, etc.) and the information about the smart card itself (i.e., the expiry of the smart card). After creating I, $\mathcal{T}$ performs the following steps [19]:

1) For small values j, $\mathcal{T}$ calculates v_j using a hash function h such that $v_j = h(I, j)$.
2) $\mathcal{T}$ chooses n distinct values of j, where v_j is quadratic residue modulo m and calculate for each chosen j the secrets s_j where s_j is the minimum solution of $s_j^2 = v_j^{-1} \mod m$.
3) $\mathcal{T}$ issues the smart card which contains I, chosen j and s_j values.

Now $\mathcal{T}$ sends the information to $\mathcal{P}$ and thus $\mathcal{P}$ can use this information for identify itself to $\mathcal{V}$. To check the identity I of $\mathcal{P}$, $\mathcal{V}$ asks $\mathcal{P}$ to calculate an equation that can only be solved when $\mathcal{P}$ knows all values of s_j. Without limiting the generality of the scheme, we will assume for the following identification protocol that $j \in [n]^+$. The identification schema between prover $\mathcal{P}$ and verifier $\mathcal{V}$ can be formally described as follows [19]:

1) $\mathcal{P}$ sends the string I to $\mathcal{V}$.
2) $\mathcal{V}$ generates $v_j = h(I, j)$ for $j = [n]^+$.

For each $i \in [t]^+$, $\mathcal{P}$ and $\mathcal{V}$ performs the following:

3) $\mathcal{P}$ chooses a random number $r_i \in [0, m)$ and sends the commitment $C_i = r_i^2 \mod m$ to $\mathcal{V}$.
4) $\mathcal{V}$ flips coin n times to generate random binary vector $(e_{i1}, \ldots, e_{in})$ and sends it to $\mathcal{P}$.
5) $\mathcal{P}$ sends $\mathcal{V}$ the proof P_i where:

$$P_i \equiv r_i \prod_{e_{ij}=1} s_j \mod m \tag{2}$$

6) $\mathcal{V}$ verifies the proof by performing the following check:

$$C_i \stackrel{?}{\equiv} P_i^2 \prod_{e_{ij}=1} v_j \mod m \tag{3}$$

As we can see in the first step of the identification scheme, $\mathcal{P}$ sends the information it wants to prove to $\mathcal{V}$. After that $\mathcal{V}$ will generate the hashes similar to what has been performed by the trust center $\mathcal{T}$. In the third step, $\mathcal{P}$ sends a quadratic residue number C_i in regard to modulo m and sends it to $\mathcal{V}$ as part of the proof in step i. Afterward, $\mathcal{V}$ determines which values of s_j and v_j that shall be used in eqs. (2) and (3) by flipping coins. $\mathcal{P}$ then calculates the proof by multiplying the value r_i it used with the product of s_j that was chosen from flipping coins. To verify $\mathcal{P}$'s proof, $\mathcal{V}$ needs to check whether the value C_i matches the square of P_i multiplied with v_j that chosen from flipping coin process. The third step until the sixth step is repeated for t times.

The second scheme mentioned in the [19] is the signature scheme. In contrast to the first scheme, this scheme does not require prover $\mathcal{P}$ and verifier $\mathcal{V}$ to repeatedly perform proofing steps. This scheme is also more secure since this scheme prevents $\mathcal{V}$ to prove even to himself that he is $\mathcal{P}$. This second scheme is also the key that is also used by some non-interactive ZKPs.

Instead of asking $\mathcal{V}$ to create random challenges for $\mathcal{P}$, this second scheme will use the result of hashing a particular message M. We know that the function h must generate a result that is indistinguishable from a random binary vector. This means that we can use the pseudo-randomness property of h to allow $\mathcal{P}$ to generate a challenge for herself.

In their paper, Fiat and Shamir divide the process into two main steps, the signing steps and the verification steps. The signing steps are performed by the prover $\mathcal{P}$ meanwhile the verification steps are performed by the verifier $\mathcal{V}$.

To create a signature of a message M, the steps performed by $\mathcal{P}$ are formally described as follows [19]:

1) For all $i \in [t]^+$, $\mathcal{P}$ chooses randoms r_i and computes the commitment $C_i = r_i^2 \mod m$.
2) $\mathcal{P}$ computes the hash $h(M, x_i, \ldots, x_t)$ and uses the first $n \cdot t$ bits to as binary vector $(e_{i1}, \ldots, e_{in})$ for $1 \leq i \leq t$. In the previous identification scheme, this binary vector was generated using a random function.
3) $\mathcal{P}$ computes for every $i \in [t]^+$ the proof P_i where:

$$P_i \equiv r_i \prod_{e_{ij}=1} s_j \mod m \tag{4}$$

4) $\mathcal{P}$ sends the identity string I, the message M, the binary vector $(e_{i1}, \ldots, e_{in})$ and all proofs P_i to $\mathcal{V}$.

Upon receiving the proof, $\mathcal{V}$ can verify the proof by performing the formally described steps as follows:

1) Similarly to the identification scheme, for all $j \in [t]^+$, $\mathcal{V}$ calculates $v_j = h(I, j)$.
2) For all $i \in [t]^+$, $\mathcal{V}$ calculates the following equation:

$$z_i \equiv P_i^2 \prod_{e_{ij}=1} v_j \mod m \tag{5}$$

3) $\mathcal{V}$ verifies the correctness of the proof when the first $n \cdot t$ bits of $h(M, z_1, \ldots, z_t)$ are $(e_{i1}, \ldots, e_{in})$.

During the verification phase, $\mathcal{V}$ checks whether the hash value from the prover $\mathcal{P}$ is equal to the hash from its value. The security here is assumed by the security of the hash function h that is used in the protocol. Thus, it is recommended to use a secure (cryptographic) hash function.

The technique of replacing the random integer from the verifier with the result from a hash function is later called the Fiat-Shamir Heuristic. With Fiat-Shamir Heuristic, the number of communication between the prover and verifier can be minimized and hence it makes the proof protocol more effective. However, it is important to correctly apply the Fiat-Shamir Heuristic, else we will have a broken proof system. In section V, we will discuss vulnerabilities which emerges from failure in correctly applying the Fiat-Shamir Heuristic.

V. The Frozen Heart Vulnerability

In April 2022, a blog post was published by Miller on a website with the name "Trail of Bits"[6]. He mentioned that there are vulnerabilities in several implementations of ZKP protocol. This vulnerability allows a malicious actor to create a false proof that is accepted by a verifier.

The Frozen Heart Vulnerability affects the implementation of non-interactive ZKP and it lies in flawed applications of the Fiat-Shamir Heuristic. The heuristic method itself is the center of the protocol that allows an interactive scheme to be used as a non-interactive protocol. Miller symbolized the Fiat-Shamir Heuristic as the heart of the protocol. The term "frozen" itself is an acronym for "FoRging Of ZEro kNowledge proofs". Hence, Miller called the vulnerability the Frozen Heart Vulnerability [6].

In this section, we will discuss protocols and open-source libraries that are affected by the Frozen Heart Vulnerability. In particular, we want to assess the weak points that introduce this weakness. We will also inspect the possible solution proposed by Miller.

A. Zk-paillier

Zk-paillier[1] is a library that contains a set of Pallier ZKP cryptosystems. It is written in Rust programming language and includes several ZKP protocol implementations, such as multiplication-mod-n^s protocol [20], range proof [21], and composite proof [22]. It also implements the discrete log with composite modulus proof based on the protocol from Girault. First, we discuss the protocol used in the implementation.

[1] https://github.com/ZenGo-X/zk-paillier

In 1991, Girault published two schemes that allow a public key to be authenticated without additional certificate [23]. The idea behind the schemes is to allow the public key calculated by both trust center $\mathcal{T}$ and the prover $\mathcal{P}$. This method allows the public key to contain information about the certificate.

The first scheme mentioned by Girault in his work is based on the RSA/Rabin digital signature scheme [24] meanwhile the other is based on the ElGamal digital signature scheme [25]. The first scheme in Girault's work mentioned two different types of protocol that build the scheme, they are identification protocol and key exchange protocol. The key exchange scheme is irrelevant for this paper and will not be discussed further because it does not use ZKP mechanism. Rather we will focus on the identification protocol.

The identification protocol of the first scheme uses the hardness of solving the discrete logarithm problem (similar to Diffie-Hellman Key Agreement protocol [26]), which is defined as follows:

Definition 5: A discrete logarithm problem is a problem where given g the generator of $(\mathbb{Z}/m\mathbb{Z})^*$ and an integer $y \in (\mathbb{Z}/m\mathbb{Z})^*$, the solver should find an integer s as solution of the following equation:

$$g^s \equiv y \mod m$$

In Girault's identification protocol, prover $\mathcal{P}$ wants to prove to verifier $\mathcal{V}$ that she knows a secret s. They use the discrete logarithm problem as the proof mechanism. $\mathcal{V}$ generates $m = p \cdot q$ where p and q are prime numbers and computes the maximal order in the multiplicative group $(\mathbb{Z}/m\mathbb{Z})^*$ as g. $\mathcal{V}$ then publishes integers m and g. The latter steps of the protocol can be formally described as follows [23], [27]:

1) $\mathcal{P}$ calculates public value $v \equiv g^{-s} \mod m$ and publishes v
2) $\mathcal{P}$ chooses a random integer $r_\mathcal{P}$ and sends commitment $C \equiv g^{r_\mathcal{P}} \mod m$ to $\mathcal{V}$
3) $\mathcal{V}$ generates a random integer $r_\mathcal{V}$ and sends it to $\mathcal{P}$
4) $\mathcal{P}$ calculates the proof $P = r_\mathcal{P} + s \cdot r_\mathcal{V}$ and sends it to $\mathcal{V}$
5) $\mathcal{V}$ checks whether C fulfills the following equation:

$$C \stackrel{?}{\equiv} g^P \cdot v^{r_\mathcal{V}} \mod m$$

As we know, Girault's proof is an interactive ZKP protocol, which means that it requires back-and-forth interaction between the prover and the verifier. In the implementation of zk-paillier, this protocol is transformed into a non-interactive ZKP protocol using Fiat-Shamir Heuristic. In comparison to its original version, the implementation uses the first t bits of the hash value of the commitment and all public values related to the proof. As a result, the protocol can be described as follows:

The prover

1) Calculates public value $v \equiv g^{-s} \mod m$ and publishes v
2) Chooses a random integer $r_\mathcal{P}$ and calculate $C \equiv g^{r_\mathcal{P}} \mod m$ as commitment
3) Calculates hash value $r_\mathcal{V} = h(g, m, v, C)$ and take first t bits from the result as replacement for random from $\mathcal{V}$ in interactive protocol
4) Calculates the proof $P = r_\mathcal{P} + s \cdot r_\mathcal{V}$
5) Sends C and P to $\mathcal{V}$

Upon receiving the values, verifier

1) Calculates $r_\mathcal{V} = h(g, m, v, P)[0{:}t]$
2) Checks whether $P \stackrel{?}{\equiv} g^P \cdot v^{r_\mathcal{V}} \mod m$ is fulfilled

In the implementation of zk-paillier library as of June 16th, 2021 (commit: d065383), v is not included in creating a hash during the third step of the prover and the first step of the verifier. This means that they only calculate $r'_\mathcal{V} = h(g, m, P)$. Including v in calculating the digest is important because its value is calculated based on secret value s, which is only known to the legitimate prover.

Failure to include v allows anyone to calculate the value $r'_\mathcal{V}$ by themselves without knowing any specific secret, since g, m are known in the protocol and x is based on a random value generated on the prover side. As a result, an attacker can construct value v without knowing s such that the value can be correctly accepted by the verifier. To find the correct value of v, Miller argued in his post that v can be solved by calculating the following [7]:

$$\begin{aligned} & C \equiv g^P \cdot v^{r'_\mathcal{V}} \mod m \\ \Leftrightarrow\ & v^{r'_\mathcal{V}} \equiv C \cdot g^{-P} \mod m \\ \Leftrightarrow\ & v \equiv \left(C \cdot g^{-P}\right)^{\left(r'^{-1}_\mathcal{V} \mod \phi(m)\right)} \mod m \end{aligned}$$

Using the above calculated v value, an attacker can use this value to trick $\mathcal{V}$ into accepting a forged proof. To fix this vulnerability, Miller suggested including the value v in calculating the digest. This ensures that the value $r_\mathcal{V}$ is always based on the value of s and thus cannot be simply calculated without knowing the secret s.

B. Bulletproof Protocol

In 2017, Bünz *et al.* published paper [4] which specifies a new ZKP scheme called Bulletproof. The proof scheme allows verifier $\mathcal{V}$ to check whether a secret value s committed by $\mathcal{P}$ lies within a certain range. Such proof is often called range proof. The Bulletproof scheme can also be combined with Multi-Party Computation (MPC) protocol to allow several nodes to jointly create a collective proof.

The Bulletproof protocol uses the hardness of discrete logarithm problems as its security measure. Similar to zk-paillier, this scheme can be made non-interactive using Fiat-Shamir Heuristic. In their work, Bünz *et al.* also specified a method to convert their interactive protocol into non-interactive ZKP protocol in their paper. However, their implementation was flawed and created a vulnerability which will be discussed later. Before discussing the mistake in the protocol, we will introduce briefly the used proof protocol.

The range proof in the Bulletproof protocol is based on Pedersen commitment protocol. They defined the Perdesen commitment as in definition 6.

Definition 6: Let $\mathbb{G}_p$ cyclic group of a prime order p. Additionally, let $y, z \in \mathbb{G}_p$ any random number in $\mathbb{G}_p$. For input committed message $M \in \mathbb{Z}_p$ and random integer $r \in \mathbb{Z}_p$, a Perdesen commitment C is defined by following equation [4]:

$$C = y^M \cdot z^r$$

Using the Pedersen commitment, the range proof protocol proposed is aimed to convince verifier $\mathcal{V}$ that a prover $\mathcal{P}$ committed to a number s within a certain range of $[0, 2^n - 1]$ for any given n. The proof can be constructed in the following formal equation:

$$\{(y, z \in \mathbb{G}, C, n; s, \gamma \in \mathbb{Z}_p) : \\ C = y^s \cdot z^\gamma \wedge s \in [0, 2^n - 1]\}$$

To properly prove the commitment C, Bünz *et al.* constructed the following protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$ with secret s, which will be the base of the Bulletproof protocol:

1) For input s and γ, $\mathcal{P}$ calculates a vector $\mathbf{a}_L \in \{0,1\}^n$ such that $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = s$ and $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$. Here $\mathbf{a}_L$ is the bit representation of s.
2) $\mathcal{P}$ chooses another random integer $r_{\mathcal{P}2} \in \mathbb{Z}_p$ and calculates $x_1 = z^{r_2} \cdot \mathbf{y}^{\mathbf{a}_L} \cdot \mathbf{z}^{\mathbf{a}_R} \in \mathbb{G}_p$
3) $\mathcal{P}$ generates randomly blinding vectors $\mathbf{b}_L, \mathbf{b}_R \in \mathbb{Z}_p^n$
4) $\mathcal{P}$ chooses random integer $r_{\mathcal{P}3} \in \mathbb{Z}_p$ and calculates $x_2 = z^{r_{\mathcal{P}3}} \cdot \mathbf{y}^{\mathbf{b}_L} \cdot \mathbf{z}^{\mathbf{b}_R} \in \mathbb{G}_p$
5) $\mathcal{P}$ sends x_1 and x_2 to $\mathcal{V}$

At this point, let us introduce two different vectors polynomials $l(X), r(X)$ and a polynomial of degree two $t(X)$ that will be used for further step [4]:

$$l(X) = (\mathbf{a}_L - r_{\mathcal{V}2} \cdot \mathbf{1}^n) + \mathbf{b}_L \cdot X \\ r(X) = \mathbf{r_{\mathcal{V}1}}^n \circ (\mathbf{a}_R + r_{\mathcal{V}2} \cdot \mathbf{1}^n + \mathbf{b}_R \cdot X) + r_{\mathcal{V}2}^2 \cdot \mathbf{2}^n \\ t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2$$

The polynomial $t(X)$ will be used by the prover $\mathcal{P}$ to convince $\mathcal{V}$ that $\mathcal{P}$ knows the full construction of $t(X)$ by allowing $\mathcal{V}$ to check on randomized points $x \in (\mathbb{Z}_p \backslash \{0\})$. Formally described, $\mathcal{P}$ and $\mathcal{V}$ performs the following steps [4]:

6) $\mathcal{V}$ generates two random challenges $r_{\mathcal{V}1}, r_{\mathcal{V}2} \in (\mathbb{Z}_p \backslash \{0\})$ and sends them to $\mathcal{P}$
7) $\mathcal{P}$ generates two randoms $\tau_1, \tau_2 \in \mathbb{Z}_p$, calculates $T_i = y^{t_i} \cdot z^{\tau_i}, i \in \{1, 2\}$, and sends T_1, T_2 to $\mathcal{V}$
8) $\mathcal{V}$ generates random challenge $r_{\mathcal{V}3} \in (\mathbb{Z}_p \backslash \{0\})$ and sends $r_{\mathcal{V}3}$ to $\mathcal{P}$
9) $\mathcal{P}$ calculates following values and sends them to $\mathcal{V}$

$$\mathbf{l} = l(r_{\mathcal{V}3}) = (\mathbf{a}_L - r_{\mathcal{V}2} \cdot \mathbf{1}^n) + \mathbf{b}_L \cdot r_{\mathcal{V}3} \\ \mathbf{r} = r(r_{\mathcal{V}3}) \\ = \mathbf{r_{\mathcal{V}1}}^n \circ (\mathbf{a}_R + r_{\mathcal{V}2} \cdot \mathbf{1}^n + \mathbf{b}_R \cdot r_{\mathcal{V}3}) + r_{\mathcal{V}2}^2 \cdot \mathbf{2}^n \\ \hat{t} = t(r_{\mathcal{V}3}) = \langle \mathbf{l}, \mathbf{r} \rangle \\ \tau_{r_{\mathcal{V}3}} = \tau_2 \cdot r_{\mathcal{V}3}^2 + \tau_1 \cdot r_{\mathcal{V}3} + r_{\mathcal{V}2}^2 \cdot \gamma \\ \mu = r_{\mathcal{P}2} + r_{\mathcal{P}3} \cdot r_{\mathcal{V}3}$$

Now $\mathcal{V}$ needs to check that $\mathbf{l} = l(r_{\mathcal{V}3})$, $\mathbf{r} = r(r_{\mathcal{V}3})$, and $t(r_{\mathcal{V}3}) = \langle \mathbf{l}, \mathbf{r} \rangle$. In order to do that, $\mathcal{V}$ needs to verify statement

$$t_0 = s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})$$

where

$$\delta(r_{\mathcal{V}1}, r_{\mathcal{V}2}) = (r_{\mathcal{V}2} - r_{\mathcal{V}2}^2) \cdot \langle \mathbf{1}^n, \mathbf{r_{\mathcal{V}1}}^n \rangle - r_{\mathcal{V}2}^3 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle$$

To solve this, $\mathcal{V}$ uses $\mathbf{z}' = \mathbf{z}^{(\mathbf{r_{\mathcal{V}1}}^{-n})}$ to replace the $\mathbf{z}$ in steps 2) and 4) and perform following steps [4]:

10) $\mathcal{V}$ calculates $z_i' = z_i^{(r_{\mathcal{V}1}^{-i+1})}$ for all $i \in [1, n]$, and thus

$$\mathbf{z}' = \left(z_1, z_2^{(r_{\mathcal{V}1}^{-1})}, \cdots, z_n^{(r_{\mathcal{V}1}^{-n+1})}\right)$$

11) $\mathcal{V}$ checks the commitment C by verifying the correctness of equation

$$y^{\hat{t}} \cdot z^{\tau_{r_{\mathcal{V}3}}} \stackrel{?}{=} C^{r_{\mathcal{V}2}^2} \cdot y^{\delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})} \cdot T_1^{r_{\mathcal{V}3}} \cdot T_2^{r_{\mathcal{V}3}^2}$$

12) $\mathcal{V}$ verifies the commitment to $\mathbf{l}$ and $\mathbf{r}$ by testing the following equation

$$x_1 \cdot x_2^{r_{\mathcal{V}3}} \cdot \mathbf{y}^{-r_{\mathcal{P}3}} \cdot (\mathbf{z}')^{r_{\mathcal{P}3} \cdot \mathbf{r_{\mathcal{V}1}}^n + r_{\mathcal{P}3}^2 \cdot \mathbf{2}^n} \stackrel{?}{=} z^\mu \cdot y^{\mathbf{l}} \cdot (\mathbf{z}')^{\mathbf{r}}$$

13) $\mathcal{V}$ checks the correctness of $\hat{t}$ by solving following equation

$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle$$

14) If all checks on step 11) to 13) are successful, $\mathcal{V}$ accepts the proof or repeats from step 8), else it rejects the proof.

Based on their paper [4], Bünz *et al.* also specified the method to convert the previous interactive proof scheme into a non-interactive scheme. They applied the Fiat-Shamir Heuristic method and thus replaced all random values with hash values of the transcript to the point of each value. With this construction, we replace the following values in the original protocol:

$$r_{\mathcal{V}1} = h(x_1, x_2) \\ r_{\mathcal{V}2} = h(x_1, x_2, r_{\mathcal{V}1}) \\ r_{\mathcal{V}3} = h(x_1, x_2, r_{\mathcal{V}1}, r_{\mathcal{V}2}, T_1, T_2)$$

According to Miller [8], such application of Fiat-Shamir Heuristic creates vulnerability in the protocol because it does not include the commitment C. This allows a malicious actor to convince $\mathcal{V}$ that the committed value lies within certain range, despite otherwise. In his blog post, Miller specified the method to exploit this vulnerability by performing the following steps [8]:

1) Choose any $s \in [0, 2^n - 1]$ and generate random value γ
2) Calculate $\mathbf{a}_L, \mathbf{a}_R, x_1, x_2$ according to the original protocol
3) Compute the polynomial $t(X)$ and save the value of t_1 and t_2
4) Generate random number t_1' and t_2' and use them to replace t_1 and t_2 to compute T_i. Now we have $T_i = y^{t_i'} \cdot z^{\tau_i}, \forall i \in \{1, 2\}$
5) Compute $\mathbf{l}, \mathbf{r}, \hat{t}$, and μ according to the original schema

6) Compute $C = y^{s'} \cdot z^{\gamma}$ where

$$s' = s + (t_1 - t_1') \cdot \left(\frac{r_{\mathcal{V}3}}{r_{\mathcal{V}2}^2}\right) + (t_2 - t_2') \cdot \left(\frac{r_{\mathcal{V}3}^2}{r_{\mathcal{V}2}^2}\right)$$

Here we can see that the value C is recalculated with a new s', which can be outside the range $[0, 2^n - 1]$ that limits s. Our proof in appendix A shows that an attacker can use this value to trick $\mathcal{V}$ to accept our statement.

Due to the ability of the attacker to forge a false proof, this vulnerability is marked as high on the NIST National Vulnerability Database (NVD) with score of 8.1[2]. To fix this vulnerability, it is required to include all public values into the hash function, such as C. This will prevent an attacker to cater the value of C such that the proof can be forged.

VI. Conclusion

In this paper, we have discussed various topics covering ZKP concepts protocols, and vulnerabilities. In section III, we introduced a new narration to introduce the concept of ZKP without the requirement of advanced prior knowledge. We also introduced two types of ZKP, interactive and non-interactive ZKP. We continued the paper by discussing the Fiat-Shamir Heuristic technique in section IV, which allows secure conversion of interactive protocol into non-interactive ZKP protocol. Lastly, we discussed the effect when the technique is applied incorrectly in section V.

We learned that it is important in designing a ZKP protocol to have a problem or puzzle, that is hard enough to solve without any clue about the solution, yet easy enough to verify the answer. The problems in the NP space match such characteristics and can be applied as a problem for ZKP challenges. We also learned that it is important to correctly apply Fiat-Shamir Heuristic technique in protocol conversion. This also means including every public value and commitment as prevention against forged proof.

We hope that our work sparks interest among public readers about ZKP and other cryptography concepts. We also hope that this paper raises awareness in correctly applying and understanding cryptographic protocols. We expect to have further and deeper research in advancing available proofs, schema, and protocols in this area of discipline. In future works, we suggest a deeper analysis of other protocols that use different methods to implement non-interactive ZKP protocol, such as zk-SNARK and zk-STARK.

Appendix

A. Proof of The Frozen Heart Vulnerability in Bulletproof Protocol

To prove the correctness of the vulnerability concept in Bulletproof protocol, we need to simulate the correctness check on step 11) of the original schema [8]. From the right side of the equation, we obtain

$$\begin{aligned}
& C^{r_{\mathcal{V}2}^2} \cdot y^{\delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})} \cdot T_1^{r_{\mathcal{V}3}} \cdot T_2^{r_{\mathcal{V}3}^2} \\
= \; & y^{\left(s + (t_1 - t_1') \cdot \left(\frac{r_{\mathcal{V}3}}{r_{\mathcal{V}2}^2}\right) + (t_2 - t_2') \cdot \left(\frac{r_{\mathcal{V}3}^2}{r_{\mathcal{V}2}^2}\right)\right) \cdot r_{\mathcal{V}2}^2} \cdot \\
& z^{\left(\gamma \cdot r_{\mathcal{V}2}^2\right)} \cdot y^{\delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})} \cdot y^{\left(t_1' \cdot r_{\mathcal{V}3}\right)} \cdot z^{(\tau_1 \cdot r_{\mathcal{V}3})} \cdot \\
& y^{\left(t_2' \cdot r_{\mathcal{V}3}\right)} \cdot z^{\left(\tau_2 \cdot r_{\mathcal{V}3}^2\right)} \\
= \; & y^{\left(s \cdot r_{\mathcal{V}2}^2 + (t_1 - t_1') \cdot r_{\mathcal{V}3} + (t_2 - t_2') \cdot r_{\mathcal{V}3}^2\right)} \cdot y^{\delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})} \cdot \\
& y^{\left(t_1' \cdot r_{\mathcal{V}3}\right)} \cdot y^{\left(t_2' \cdot r_{\mathcal{V}3}\right)} \cdot z^{\left(\gamma \cdot r_{\mathcal{V}2}^2\right)} \cdot z^{(\tau_1 \cdot r_{\mathcal{V}3})} \cdot \\
& z^{\left(\tau_2 \cdot r_{\mathcal{V}3}^2\right)} \\
= \; & y^{\left(s \cdot r_{\mathcal{V}2}^2 + t_1 \cdot r_{\mathcal{V}3} + t_2 \cdot r_{\mathcal{V}3}^2\right)} \cdot y^{\delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})} \cdot y^{-\left(t_1' \cdot r_{\mathcal{V}3}\right)} \cdot \\
& y^{\left(t_1' \cdot r_{\mathcal{V}3}\right)} \cdot y^{-\left(t_2' \cdot r_{\mathcal{V}3}\right)} \cdot y^{\left(t_2' \cdot r_{\mathcal{V}3}\right)} \cdot \\
& z^{\left(r_{\mathcal{V}2}^2 \cdot \gamma + \tau_1 \cdot r_{\mathcal{V}3} + \tau_2 \cdot r_{\mathcal{V}3}^2\right)} \\
= \; & y^{\left(s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2}) + t_1 \cdot r_{\mathcal{V}3} + t_2 \cdot r_{\mathcal{V}3}^2\right)} \cdot \\
& z^{\left(r_{\mathcal{V}2}^2 \cdot \gamma + \tau_1 \cdot r_{\mathcal{V}3} + \tau_2 \cdot r_{\mathcal{V}3}^2\right)} \\
= \; & y^{\left(s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2}) + t_1 \cdot r_{\mathcal{V}3} + t_2 \cdot r_{\mathcal{V}3}^2\right)} \cdot z^{\tau_{r_{\mathcal{V}3}}}
\end{aligned}$$

We can see that the exponent of z on both side are the same, thus we can eliminate z on both side of the equation. We are left with the variable y, resulting

$$\begin{aligned}
& y^{\hat{t}} = y^{\left(s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2}) + t_1 \cdot r_{\mathcal{V}3} + t_2 \cdot r_{\mathcal{V}3}^2\right)} \\
\Leftrightarrow \; & \log_y\left(y^{\hat{t}}\right) = \log_y\left(y^{\left(s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2}) + t_1 \cdot r_{\mathcal{V}3} + t_2 \cdot r_{\mathcal{V}3}^2\right)}\right) \\
\Leftrightarrow \; & \hat{t} = s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2}) + t_1 \cdot r_{\mathcal{V}3} + t_2 \cdot r_{\mathcal{V}3}^2 \\
\Leftrightarrow \; & t_0 = s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})
\end{aligned}$$

Since we have shown that $t_0 = s \cdot r_{\mathcal{V}2}^2 + \delta(r_{\mathcal{V}1}, r_{\mathcal{V}2})$, we have successfully tricked $\mathcal{V}$ that our commitment C is for value s, which lies in the range $[0, 2^n - 1]$. However, our new secret s' can lie outside this range, meaning that our statement is false but $\mathcal{V}$ accepts it.

[2]https://nvd.nist.gov/vuln/detail/CVE-2022-29566

Lomaybe cite this instead?

References

[1] Apple and Google, *Exposure notification – cryptography specification v1.2*, 2020.

[2] F. Hao, P. Y. Ryan, and P. Zieliński, "Anonymous voting by two-round public discussion," *IET Information Security*, vol. 4, no. 2, pp. 62–67, 2010.

[3] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Financial Cryptography and Data Security: 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers 21*, Springer, 2017, pp. 357–375.

[4] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, *Bulletproofs: Short proofs for confidential transactions and more*, Cryptology ePrint Archive, Paper 2017/1066, version 20180701:235657, https://eprint.iacr.org/archive/2017/1066/20180701:235657, 2017. [Online]. Available: https://eprint.iacr.org/archive/2017/1066/20180701:235657.

[5] D. Bernhard, O. Pereira, and B. Warinschi, "How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios," in *ASIACRYPT*, vol. 7658, Springer, 2012, pp. 626–643. DOI: 10.1007/978-3-642-34961-4_38. [Online]. Available: https://www.iacr.org/archive/asiacrypt2012/76580619/76580619.pdf.

[6] J. Miller, *Coordinated disclosure of vulnerabilities affecting Girault, Bulletproofs, and PlonK*, Accessed on June 10th, 2023, Apr. 2022. [Online]. Available: https://blog.trailofbits.com/2022/04/13/part-1-coordinated-disclosure-of-vulnerabilities-affecting-girault-bulletproofs-and-plonk/.

[7] J. Miller, *The Frozen Heart vulnerability in Girault's proof of knowledge*, Accessed on June 10th, 2023, Apr. 2022. [Online]. Available: https://blog.trailofbits.com/2022/04/14/the-frozen-heart-vulnerability-in-giraults-proof-of-knowledge/.

[8] J. Miller, *The Frozen Heart vulnerability in Bulletproofs*, Accessed on June 10th, 2023, Apr. 2022. [Online]. Available: https://blog.trailofbits.com/2022/04/15/the-frozen-heart-vulnerability-in-bulletproofs/.

[9] J. Miller, *The Frozen Heart vulnerability in PlonK*, Accessed on June 10th, 2023, Apr. 2022. [Online]. Available: https://blog.trailofbits.com/2022/04/18/the-frozen-heart-vulnerability-in-plonk/.

[10] M. Petkus, *Why and how zk-snark works*, 2019. arXiv: 1906.07221 [cs.CR].

[11] WIRED, *Computer scientist explains one concept in 5 levels of difficulty — wired*, *YouTube*, Accessed on June 10th, 2023, Jan. 2022. [Online]. Available: https://www.youtube.com/watch?v=fOGdb1CTu5c.

[12] J.-J. Quisquater, M. Quisquater, M. Quisquater, *et al.*, "How to explain zero-knowledge protocols to your children," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed., New York, NY: Springer New York, 1990, pp. 628–631, ISBN: 978-0-387-34805-6.

[13] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 203–225.

[14] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

[15] B. Kaliski, "Quadratic residue," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 1003–1003, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_428. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_428.

[16] B. Kaliski, "Legendre symbol," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 716–716, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_418. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_418.

[17] B. Kaliski, "Jacobi symbol," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 655–655, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_416. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_416.

[18] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 47–53, ISBN: 978-3-540-39568-3.

[19] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194, ISBN: 978-3-540-47721-1.

[20] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4*, Springer, 2001, pp. 119–136.

[21] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 431–444, ISBN: 978-3-540-45539-4.

[22] C. Hazay, G. L. Mikkelsen, T. Rabin, T. Toft, and A. A. Nicolosi, "Efficient RSA key generation and threshold paillier in the two-party setting," *Journal of Cryptology*, vol. 32, no. 2, pp. 265–323, 2019.

[23] M. Girault, "Self-certified public keys," in *Advances in Cryptology — EUROCRYPT '91*, D. W. Davies, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 490–497, ISBN: 978-3-540-46416-7.

[24] M. O. Rabin, "Digitalized signatures and public key functions as intractable as factorization," MIT Laboratory for Computer Science, Tech. Rep., 1979.

[25] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985. DOI: 10.1109/TIT.1985.1057074.

[26] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976. DOI: 10.1109/TIT.1976.1055638.

[27] D. Pointcheval, "The composite discrete logarithm and secure authentication," in *Public Key Cryptography*, H. Imai and Y. Zheng, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 113–128, ISBN: 978-3-540-46588-1.