International Journal of Information Security
https://doi.org/10.1007/s10207-023-00779-8

REGULAR CONTRIBUTION

Enhancing security in Fiat–Shamir transformation-based non-interactive zero-knowledge protocols for IoT authentication

Firas Hamila[1] · Mohammad Hamad[1] · Daniel Costa Salgado[2] · Sebastian Steinhorst[1]

Abstract
With the rapid expansion of IoT devices and their applications, there is an increasing demand for efficient and secure authentication mechanisms to protect against unauthorized access. Traditional authentication mechanisms face limitations regarding computational speed, communication costs, and vulnerability to cyber-attacks. Zero-knowledge proof (ZKP) protocols have emerged as an effective solution for achieving secure and efficient authentication in such environments without revealing sensitive information. Among ZKP protocols, Σ-protocols, a class of interactive ZKP protocols, have been employed for their efficiency and security. However, their interactive nature necessitates multiple rounds of communication, which can reduce efficiency and increase communication overhead for resource-constrained devices. Many works have aimed to eliminate the interaction of Σ-protocols by utilizing a transformation called the Fiat–Shamir transformation (FST). However, there is still a concern regarding the soundness of the FST as it can sometimes convert a secure Σ-protocol into an insecure non-interactive zero-knowledge (NIZK) authentication scheme. In this paper, we propose an approach for transforming Σ-protocols into a NIZK protocol based on the FST, yielding significant enhancements in efficiency, communication overhead reduction, and elimination of interaction. Our proposed protocol enables the completion of the authentication process in a single request while also strengthening the soundness of Σ-protocols in comparison with the traditional FST by requiring two authentication factors instead of one. To demonstrate our approach's robustness, we conducted comprehensive informal and formal security analyses (using the Tamarin-Prover). Our protocol demonstrated completeness, soundness, zero-knowledge properties, and robustness against attacks, including eavesdropping, message modification, replay, and brute force attacks. Additionally, our performance analysis displayed a remarkable 50% improvement in computational cost compared to traditional Σ-protocols, underscoring its efficiency for practical use.

Keywords Authentication · IoT · Non-interactive zero-knowledge proof · Σ-Protocols · Fiat–Shamir transformation

1 Introduction

The rapid expansion of Internet of Things (IoT) has transformed our interaction with the environment. These devices have become an integral part of our daily routines, offering convenience, efficiency, and enhanced productivity. However, the exponential adoption of IoT across various domains has also brought forth significant concerns surrounding the security and privacy of these ecosystems. To ensure secure interaction among IoT devices, a crucial element is the establishment of robust authentication mechanisms. However, authentication in IoT poses several challenges due to the resource limitations of IoT devices, such as their limited computational capabilities, memory, and power sources. Consequently, traditional authentication mechanisms prove impractical and susceptible to various attacks.

Many approaches have been developed to address the need for secure authentication of IoT communication. These approaches vary in their methodologies and cryptographic techniques, including pre-shared key (PSK)-based authentication [39], certificate-based authentication [38], public key

✉ Mohammad Hamad
mohammad.hamad@tum.de

Firas Hamila
firas.hamila@tum.de

Daniel Costa Salgado
daniel.salgado@exxeta.com

Sebastian Steinhorst
sebastian.steinhorst@tum.de

1 Technical University of Munich, Munich, Germany

2 Exxeta AG, Mannheim, Germany

Published online: 25 November 2023

infrastructure (PKI)-based authentication [37], token-based authentication [19], and one-time password authentication [27]. However, these approaches suffer from limitations. For example, some require the storage of large amounts of pre-calculated authentication keys [27], while others involve the complex management and distribution of tokens and certificates [19, 38]. Furthermore, there is a security risk associated with the potential compromise of authentication keys [37, 39]. Finally, many approaches do not preserve privacy, leaving personal information revealed [19, 39].

Different approaches have been proposed to overcome the limitations of traditional authentication mechanisms regarding privacy. One of these approaches is the ZKP protocol [15]. These protocols enable one party, known as the *prover*, to demonstrate the validity of a statement to another party, known as the *verifier*, without disclosing any information beyond the truth of the statement itself. This characteristic makes ZKP protocols appealing for authentication scenarios, as they enable authenticated and private communications. Many ZKP techniques require a large number of interactive rounds to authenticate a prover [15], introducing more computational overhead, while others need a reduced interaction [17].

One class of interactive ZKP, known as Σ-protocols [17] (see Sect. 2.1), provides a secure and efficient method for proving knowledge of a witness, such as a secret key or any valid credential, without actually revealing that information. These protocols are more efficient than traditional ZKPs since they utilize built-from-scratch cryptographic functions [41]. However, Σ-protocols still require 3-round interactions between the prover and the verifier to prove knowledge of a secret. Recently, many research studies have been centered on the advancement of efficiency and security of ZKPs [43]. One of these research works was focused on utilizing a transformation called FST [24] (see Sect. 2.4) to convert Σ-protocols into non-interactive ZKP protocols. The resulting NIZK protocols have been considered the most efficient NIZK protocol [6, 30] and more efficient and straightforward than the ones based on cryptographic primitives [31]. All of these factors contribute to the high suitability of FST-based NIZK protocols for deployment in IoT scenarios.

While significant progress has been made in the adoption of Σ-protocols and FST-based authentication techniques for IoT systems, there is still a concern regarding the soundness of the FST as it can sometimes turn a secure Σ-protocol into an insecure NIZK authentication scheme [26]. Many research works argued that security properties of FST are hard to achieve, even though it seems to produce secure schemes in practice when applied to the correct protocols [9, 22, 26]. To improve the soundness of this heuristic, extensive research has been conducted, exploring various approaches. These include investigating specific groups of hash functions [31], imposing restrictions on permitted Σ-protocols [36], and considering encryption as an alternative to the employed hash function [18]. However, *it is important to note that a comprehensive solution that effectively enhances the soundness of the FST and the employed protocol across all types of hash functions and Σ-protocols while maintaining satisfactory performance levels remains unexplored.* In light of this problem, our paper aims to contribute to the field by proposing an approach that tackles these challenges and offers an improved solution.

1.1 Contribution

In this paper, we propose an efficient and secure transformation from interactive 3-round Σ-protocols to NIZK protocols. Our approach is based on improving the soundness of the FST for *all classes* of secure Σ-protocols. Specifically, we propose a non-interactive transformation based on the FST that requires two verifications with two distinct keys for approval. Our protocol not only verifies the validity of the ZKP but also verifies the authenticity of the challenge by utilizing a secondary key. Our proposed solution strengthens the security guarantees of the transformed protocols by adding an additional layer of protection against potential attacks. Our proposed protocol mitigates the risks associated with applying the FST to Σ-protocols, as mentioned previously, by requiring two verifications with different keys. This enhancement significantly improves the efficiency and the overall robustness and soundness of the authentication process, ensuring that only legitimate entities gain access, regardless of the Σ-protocol employed. Furthermore, we extend our contribution beyond theoretical advancements by providing a practical implementation and a performance analysis of the proposed authentication protocol in the programming language Rust. We also leverage the capabilities of the Tamarin-prover [40], a tool for analyzing security protocols, to model and evaluate the security properties of our NIZK protocol. Our contribution lies in demonstrating how ZKP protocols based on non-primitive cryptographic systems, such as the Schnorr Identification Scheme (SIS) (see Sect. 2.3), can be modeled and analyzed using Tamarin-prover, expanding the applicability of this tool in the domain of IoT security. It is important to note that there is currently no existing assessment of the security properties of similar non-interactive protocols based on non-primitive cryptographic functions in the literature.

The main contributions of this article are as follows:

- We propose a NIZK protocol that requires two verifications with two distinct keys, enhancing the security and soundness of the FST and the utilized Σ-protocols, while improving its efficiency (see Sect. 4).

- We utilize the Tamarin-prover to model and formally analyze the security properties of the proposed NIZK protocol (see Sect. 5.2).
- We demonstrate the feasibility and applicability of our proposed protocol by providing an open source[1] implementation of it using the Rust programming language and evaluating its performance on embedded resource-constrained platforms (see Sect. 5.3).

2 Background

In this section, we provide an overview of the basic concepts that our proposed solution was built upon. First, Sect. 2.1 demonstrates the basics of Σ-protocols. Then, the elliptic curve discrete logarithm problem is introduced in Sect. 2.2. Next, Sect. 2.3 depicts the Schnorr identification scheme. After that, Sect. 2.4 characterizes the Fiat–Shamir transformation, focusing on its efficiency and soundness. Finally, in Sect. 2.5, the elliptic curve Diffie–Hellman ephemeral key agreement protocol (ECDHE) is introduced.

2.1 Σ-Protocols

Σ-protocols are identification schemes based on a 3-round interactive ZKP. They generally work as follows: for a proof system (A, B), where A is the prover and B the verifier, $(x, w) \in R$ where R is a relation such as w is a witness of a statement x known only by A. To prove that $x \in R$:

1. The prover A sends a message a, called commitment, to the verifier B.
2. After receiving a, B sends back a t-bit random c, called the challenge.
3. A sends a response y, called the proof, based on the commitment a, the challenge c, and the witness w. B accepts or rejects the proof of A based on the received information.

To define Σ-protocols, assuming that the prover A, and the verifier B, are both probabilistic polynomial time machines, where only A knows w, a protocol is called Σ-protocol for a relation R, where $(x, w) \in R$ if the following 4 properties hold [17]:

- The protocol has a 3-round interaction between A and B, as described above.
- Completeness: The verifier B always accepts $\forall(x, w) \in R$.
- Special Soundness: the proof can only be accepted by the verifier B if the statement being proven is actually correct, meaning $(x, w) \in R$.
- Honest-verifier zero-knowledge: the proof can only be accepted if both the prover A and verifier B are following the protocol honestly without cheating, for example, having a true random number generator is considered honesty.

Σ-protocols rely on two key properties [17]. Firstly, the relation R must be a hard relation, making it computationally difficult to compute w from x, while efficiently generating (x, w). Secondly, the witness-hiding (WH) property, a subset of zero-knowledge, ensures that the system provides privacy and security for the prover by keeping the witness w hidden. There are several implementations of Σ-protocols. One example is Schnorr's protocol [17].

2.2 Elliptic curves discrete logarithm problem (ECDLP)

The discrete logarithm problem (DLP) can be defined over elliptic curves [33]. Let $\mathcal{E}$ be an elliptic curve (EC). P and Q are points on $\mathcal{E}$, with P being a generator $(P,\ Q \in \mathcal{E})$. The ECDLP is the task of finding the integer s such that:

$$Q = sP \tag{1}$$

Finding s is computationally infeasible. The multiplication, in this case, denotes the multiplication of a scalar with a point on the elliptic curve, which is different from the traditional multiplication as it is computationally infeasible to compute the scalar s by just knowing the points Q and P.

2.3 Schnorr's identification scheme (SIS)

SIS is a Σ-protocol used to prove knowing a secret without revealing its value [17]. SIS is considered more efficient and less complex in terms of computation and communication compared to primitive cryptography-based Identification Schemes, such as the challenge-and-response protocol in the public key setting. This efficiency is achieved because SIS's building blocks do not rely on any additional cryptographic methods [41].

One way to implement SIS is to build it over EC based on the ECDLP. For an EC, $\mathcal{E}$, P and Q are points on $\mathcal{E}$, with P being a generator and Q is defined as in 1. The prover A's private key is s, while P and Q form his public key. To prove his knowledge of the private key s, A interacts with the verifier B using a 3-round interaction, as follows [28]:

1. A commits to a random integer r and sends the commitment R to B such that:

$$R = rP \tag{2}$$

[1] Link: https://github.com/tum-esi/act-nizkp.

2. B responds by sending a random challenge number c.
3. A sends the response y to B such that:

$$y = r + cs \tag{3}$$

The verifier B accepts the identification only if:

$$yP = R + cQ \tag{4}$$

The security of the SIS over EC originates from the ECDLP and relies on the presumed intractability of calculating discrete logarithms in the chosen EC group, meaning that the EC has to be chosen carefully to maintain the security of the SIS [28].

2.4 The Fiat–Shamir transformation (FST)

FST presents simple and secure non-interactive signature and identification schemes that are suitable for low-power devices [24]. The idea behind it was to transform any interactive 3-round protocol that utilizes the challenge and response method into a non-interactive form by replacing the challenge with a publicly verifiable hash function, such as SHA-256, SHA-384, or SHA-512. The challenge is usually sent from the verifier to the prover to ensure the proof's randomness and soundness. The security of this transformation is related to the assumption of the existence of a secure cryptographic Hash function that can act as a random oracle model.[2] The FST has been recognized as one of the most efficient non-interactive protocols [6, 30, 31].

2.4.1 The security of the FST

Several studies have been conducted to prove the security of this efficient transformation to non-interactive protocols. Canetti et al. [6] studied two slightly different variants of the FST to compare their security properties. Their findings demonstrated that the variant where both the commitment and the statement need to be hashed provides better security guarantees under the random oracle assumption compared to the variant where only the commitment needs to be hashed. Faust et al. [23] conducted a study to evaluate the security of the transformation against attacks where the adversary modifies a valid proof into another proof that would be accepted by the verifier. In the random oracle model, they proved that the FST is secure against such types of attacks.

Despite the security claims of the Fiat–Shamir transformation in the random oracle model, recent studies have shown that it might not be as secure as claimed. Goldwasser et al. [26] examined the security of this transformation and disproved its security, contrary to what some other papers argued. They proved that in certain scenarios, the FST fails to maintain the soundness of the protocol, depending on the employed Σ-protocol. They demonstrated that even if the used hash function is deemed secure and collision-resistant, there exists a secure Σ-protocol that loses its security when the FST is applied. This occurs because certain Σ-protocols can lose their security when the verifier's challenge is replaced by a hash. Dwork et al. [22] referred to the hash function required by the FST as a "magic function" due to its challenging security requirements. They conducted a study to investigate the existence of such a function that would ensure the resulting non-interactive protocol remains secure. Their study demonstrated that if a 3-round proof system is considered to be an ultra-weak zero-knowledge protocol,[3] and it is used to construct a non-interactive signature scheme using the FST, then there exists a forger who can successfully forge a signature with a high probability. This result does not imply that every 3-round interactive ZKP becomes insecure when the FST is applied. However, it shows that replacing the verifier's challenge with a hash function can lead to insecure protocols, especially those that meet the requirements of an ultra-weak zero-knowledge proof system.

2.5 ECDHE key agreement protocol

ECDHE is a method that allows two different parties to agree upon a shared secret key over an insecure channel. This is done by using the computational difficulty of the ECDLP [1]. ECDHE works as follows: Given an elliptic curve $\mathcal{E}$ and P is a generator, two parties A and B can generate a shared secret key using the following steps:

- A chooses a random secret integer d_A. He computes
$$Q_A = d_A P \tag{5}$$
and sends Q_A to B.
- B also generates a random secret integer d_B and computes
$$Q_B = d_B P \tag{6}$$
and sends Q_B to A.
- Now A can compute $d_A Q_B$ and B can compute $d_B Q_A$, which results in the same shared secret key SK_{AB}, such that:

$$SK_{AB} = d_A d_B P = d_B d_A P \tag{7}$$

[2] In cryptography, the Random Oracle Model is a concept that provides a way to evaluate the security of cryptographic protocols by assuming that a hash function behaves like a black box and generates uniformly distributed and unpredictable outputs for any input it receives.

[3] An ultra-weak ZKP is a protocol that has been intentionally weakened to reduce its security [22].

An attacker can intercept the Q_A and Q_B values, but they cannot compute the shared secret key without solving the ECDLP. However, this implementation is still not secure against man-in-the-middle attacks [1]. For that reason, an authenticated key agreement has to be used instead. In this case, we can do it by combining SIS over EC with the ECDLP.

3 System and threat model

Our proposed authentication system is for device-to-device communication in a mesh network of nodes $\{N_1, N_2, N_3, \ldots\}$ where each node can communicate with any other node. In our system, P is a public point on the elliptic curve $\mathcal{E}$, which is used as a generator. Each node N_i has:

- s_{N_i} is a private key of the node N_i.
- Q_{N_i} is a public key of node N_i, such that $Q_{N_i} = s_{N_i} P$.

In our system, we assume the existence of a trusted authority (TA) that can use a traditional Public Key Infrastructure (PKI) or a distributed one to sign and share the public keys. However, the specific interpretation and implementation of such a TA are beyond the scope of this paper. Additionally, each node is capable of utilizing the SHA-3 and the Keccak Message Authentication Code (KMAC) functions. Secure Hash Algorithm 3 (SHA-3) represents the Keccak hash function, which was chosen as the new standard by National Institute of Standards and Technology (NIST) in 2012 [8]. And KMAC defines theMessage Authentication Code (MAC) algorithm based on the Keccak function [32]. We chose to use the SIS, the SHA-3, and the KMAC in our authentication protocol; however, our NIZK transformation applies to any Σ-protocol and to any secure hash and MAC functions.

3.1 General security assumptions

In this section, we will illustrate the general security assumptions that our protocol is based on. The strength of these assumptions provides a strong foundation for the security of our NIZK IoT device-to-device authentication protocol.

1. ECDHE security assumption: The security of the authenticated ECDHE protocol is based on the computational Diffie–Hellman assumption in elliptic curve groups, which implies that it is computationally infeasible for an attacker to compute the shared secret key between two devices based on the public information exchanged during the protocol.
2. SIS security assumption: The security of the SIS over EC is based on the computational Diffie–Hellman assumption in elliptic curve groups and the intractability of the discrete logarithm problem, which indicates that it is computationally infeasible for an attacker to forge a valid schnorr proof without knowing the private key of the device. Assuming the difficulty of the elliptic curve discrete logarithm problem, the SIS has been proven to satisfy the following properties of a zero-knowledge protocol (i.e., completeness, soundness, and honest-verifier zero-knowledge).
3. MAC security assumption: The security of the MAC function used to replace the challenge of the verifier in the NIZK transformation is based on the unforgeability of the underlying MAC algorithm, which means that it is computationally infeasible for an attacker to generate a valid MAC tag without knowing the secret key. Additionally, the MAC function should output a different random-looking tag for each message and key. To be more precise, we use the KMAC function in our protocol, which signifies that the KMAC function has to be collision resistant, unforgeable, and provide randomness.

3.2 Threat model

In this section, we will illustrate the threat model considered in our formal security analysis using Tamarin-prover (more information about Tamarin-prover and the formal security analysis is available in Sect. 5.2). Our threat model is based on the Dolev–Yao adversary model [20], where potential attacks and vulnerabilities can be exploited by an adversary attempting to compromise the system's security. The Dolev–Yao model assumes an insecure and untrusted network with an adversary that can intercept, modify, and generate arbitrary messages, meaning that he can impersonate users and eavesdrop on the communication channel to gather sensitive information. Additionally, we also make the following assumptions:

- We consider cryptographic functions, such as hash and MAC functions, as perfect, meaning that generating a valid MAC tag requires knowing the secret key.
- We assume that the nodes are honest, which implies that they have perfect random number generators. The nodes will faithfully execute the protocol steps, correctly generate and transmit messages, and respect the security guarantees and requirements of the protocol.
- We also consider that the long-term private keys are stored securely and cannot be compromised through attacks such as side-channel attacks.

3.3 Attack vectors

In this section, we present various attack vectors that have been considered in the evaluation of our security protocol. These attack vectors include:

3.3.1 Man-in-the-middle (MITM) attack

In the MITM attack, the intruder positions himself between two nodes and intercepts their communication, giving him the ability to eavesdrop on the conversation or manipulate the communication flow. In this paper, we distinguish between several types of attacks that can be considered part of MITM: eavesdropping, message modification, and replay attacks.

- Active attacks: the adversary actively participates in the communication between the prover and the verifier. Their objective is to obtain valuable login information or credentials during the interaction. This acquired information can then be used by the adversary at a later time to impersonate the prover to the verifier.
- Eavesdropping: The adversary passively listens to the conversation between the prover and the verifier, attempting to gather sensitive information, such as secret keys and other confidential data, that can be used to impersonate the prover to the verifier.
- Message modification attacks: The adversary intercepts a message during transmission between the prover and the verifier and makes changes to its content, aiming to forward a falsified message.
- Replay attacks: the adversary captures and records the data transmissions between the prover and the verifier and subsequently replays those same transmissions back to one or both parties at a later time without initiating a separate session. This allows the adversary to reuse the captured values to authenticate themselves successfully. By reusing the data, the adversary can impersonate one of the parties.

3.3.2 Brute force attack

The adversary employs different approaches in which they repeatedly guess the correct authentication keys. This attack involves systematically checking all possible combinations until the adversary successfully cracks the correct key. It can be executed as a dictionary attack, where pre-defined values are used, or as a blind attack, where the intruder tries random values until the correct one is found. While this is a straightforward attack, it can be highly time-consuming.

3.4 System requirements

Our proposed NIZK protocol should fulfill the essential properties that define a ZKP: completeness, soundness, and zero-knowledge.

4 Proposed solution

Our proposed protocol is based on utilizing a NIZK transformation on a Σ-protocol by applying the FST to the SIS over EC. However, our approach enhances prior work by incorporating an extra layer of security through the utilization of a keyed hash function instead of a standard hash function in the FST. The shared secret key used in the transformation is dynamically communicated through a one-time interactive mutual authentication with ECDHE key agreement protocol during the *Setup Phase*, as described in Sect. 4.1. This shared key will be updated after each authentication process. To illustrate the two steps of the protocol, we will consider a scenario involving two nodes: the prover (referred to as A) and the verifier (referred to as B).

4.1 Setup phase

The goal of this phase is to establish a shared secret key that will be utilized for the keyed hash function in the NIZK transformation. It is important to note that this phase needs to be executed once during the initial communication between two nodes. During this phase, any secure key-sharing protocol can be employed. In this work, we utilize the authenticated ECDHE protocol to share a secret key A and B. The authentication is performed using the interactive 3-round SIS protocol. The steps of this process are illustrated in Fig. 1 and work as follows:

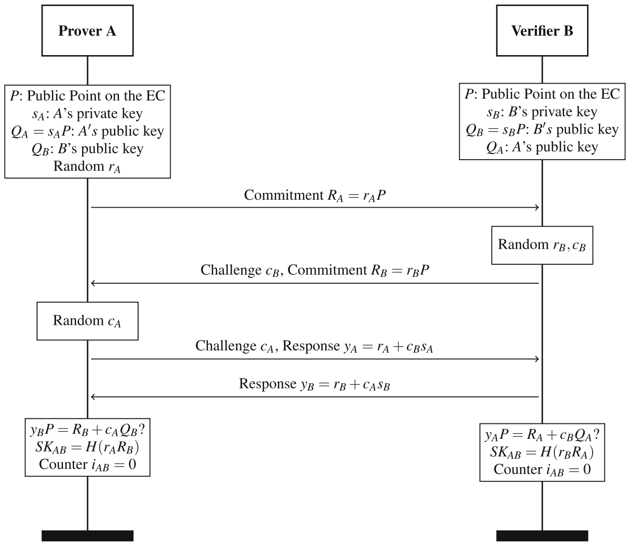

Fig. 1 Setup phase: utilizing ECDHE for secure SK_{AB} sharing between the prover and the verifier

1. A commits to B by generating the commitment R_A such that:

$$R_A = r_A P \tag{8}$$

   where r_A is a random integer generated by A. A sends R_A to B.
2. B also commits to A and generates the commitment R_B such as:

$$R_B = r_B P \tag{9}$$

   where r_B is a random value generated by B. B generates also another random integer c_B that he will provide as a challenge to A. He sends c_B and R_B back.
3. A generates a random integer challenge c_A, and the response y_A such that:

$$y_A = r_A + c_B s_A \tag{10}$$

   and sends c_A and y_A to B.
4. B responds with his response y_B, such that:

$$y_B = r_B + c_A s_B \tag{11}$$

5. Now both A and B, verify the authentication of each other using the values they received, such that:

$$y_A P = R_A + c_B Q_A, \quad y_B P = R_B + c_A Q_B \tag{12}$$

   If true: A calculates the secret key SK_{AB}, such that:

$$SK_{AB} = H(r_A R_B) \tag{13}$$

   and B calculate it, such that:

$$SK_{AB} = H(r_B R_A) \tag{14}$$

   Upon successful verification, the secret shared key is SK_{AB} is computed as determined in Eq. 15.

$$SK_{AB} = H(r_A R_B) = H(r_B R_A) = H(r_A r_B P) \tag{15}$$

   Finally, A and B also initialize a shared counter i_{AB} to 0 since it will be employed later in the NIZK proof. Hashing the secret values is important to obtain the desired key length and to ensure the randomness of the key.

4.2 NIZK transformation

4.2.1 Approach and steps

Our proposed solution aims to convert the 3-round SIS protocol into a non-interactive protocol by employing a modified version of the FST. The details of the proposed NIZK protocol are provided in Fig. 2 and assume the successful execution of the setup phase described in Sect. 4.1. When A wants to authenticate to B, A prepares all the necessary data and transmits them in a non-interactive manner. First, A generates a random integer r_A and calculates the commitment R_A from it as follow:

$$R_A = r_A P \tag{16}$$

Then, A needs to generate B's challenge, denoted as c_B. The generation of c_B can be determined using Eq. 17. To perform this step, A utilizes the counter i_{AB}, which is shared between A and B. Using the MAC function KMAC with the shared secret key SK_{AB}, A generates the MAC of the commitment R_A and the next value of the counter i_{AB} $(i_{AB} + 1)$.

$$c_B = MAC_{SK_{AB}}(R_A \mid\mid i_{AB} + 1) \tag{17}$$

After that, A uses c_B to generate his response y_A, as determined in Eq. 18.

$$y_A = r_A + c_B s_A \tag{18}$$

Finally, A sends the commitment R_A, the challenge c_B, and the response y_A all together to B.

Upon receiving this information, B verifies his challenge c_B to ensure its correct generation according to the protocol. This verification is performed using Eq. 17 with the shared

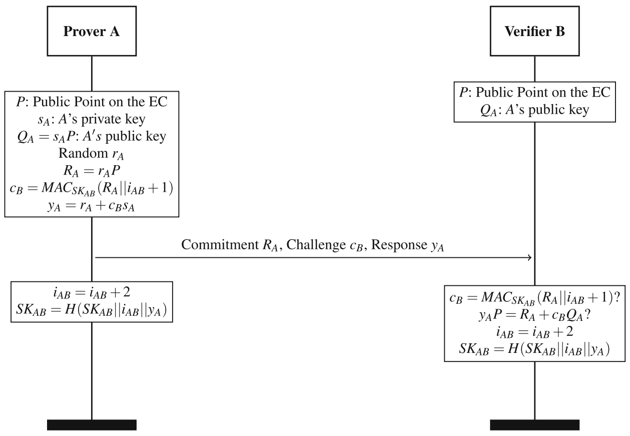
Prover A
Verifier B
P: Public Point on the EC
s_A: A's private key
$Q_A = s_A P$: A's public key
Random r_A
$R_A = r_A P$
$c_B = MAC_{SK_{AB}}(R_A||i_{AB}+1)$
$y_A = r_A + c_B s_A$
P: Public Point on the EC
Q_A: A's public key
Commitment R_A, Challenge c_B, Response y_A
$i_{AB} = i_{AB} + 2$
$SK_{AB} = H(SK_{AB}||i_{AB}||y_A)$
$c_B = MAC_{SK_{AB}}(R_A||i_{AB}+1)$?
$y_A P = R_A + c_B Q_A$?
$i_{AB} = i_{AB} + 2$
$SK_{AB} = H(SK_{AB}||i_{AB}||y_A)$

Fig. 2 Non-interactive zero-knowledge proof protocol between the prover and the verifier

secret key SK_{AB} and the incremented counter i_{AB}. If the verification succeeds, B then checks the correctness of Schnorr's proof using Eq. 19.

$$y_A P = R_A + c_B Q_A \tag{19}$$

If Eq. 19 holds, B successfully authenticates A.

After successful authentication, both A and B need to update the secret key SK_{AB} and the shared counter i_{AB} for future use. First, the counter i_{AB} is incremented and saved, ensuring that $i_{AB} = i_{AB} + 2$ (since $i_{AB} + 1$ was used in the proof and should not be reused). Then, the new shared secret key SK_{AB} is calculated as the hash of the old key SK_{AB}, the counter i_{AB}, and the response y_A, as specified in Eq. 20.

$$SK_{AB} = H(SK_{AB} \mid\mid i_{AB} \mid\mid y_A). \tag{20}$$

4.3 Protocol design considerations

Note that A can include the value of the counter i_{AB} with the proof to prevent counter value synchronization errors between A and B. However, B must always keep track of the most recently received counter value. Only requests with a received counter value greater than the last used one will be accepted. This precaution is crucial for avoiding replay attacks and ensuring the correct functioning of the protocol, even in scenarios where acknowledgment of proof verification is not employed. Also, by including y_A in the computation, a more randomized value for the new key is achieved since y_A is generated using random values, thereby ensuring its randomness as well.

Practically, the counter i_{AB} can be implemented as a 32bit unsigned integer that starts at value 0. When nearly all the possible values of i_{AB} are used, A and B have to initiate a mutual authentication to agree upon a new shared secret key. Then, i_{AB} could be reset to 0 and used again securely without the risk of reply attacks. This procedure is illustrated in Algorithm 1.

In addition, the messages sent using our NIZK proof are authenticated, ensuring integrity and authentication. Our NIZK transformation, when used without mutual authentication, primarily guarantees message integrity and the authentication of the prover. For our implementation, we opted to use the KMAC function due to its favorable performance and security against cyber-attacks. However, alternative MAC functions can also be employed for the same purpose.

Algorithm 1: Shared Counter Reset

```
Input: $i_{AB}$: Shared counter between $A$ and $B$, $SK_{AB}$: Shared
       secret key between $A$ and $B$
if $i_{AB} > 2^{32} - 4$ then
    Agree upon a new $SK_{AB}$;
    $i_{AB} = 0$;
else
    $i_{AB} = i_{AB} + 2$;
```

Using a keyed hash function with a counter instead of an unkeyed hash, as in the FST, ensures that the challenge is always *random, unique, and authenticated by the verifier*. Our solution addresses the issue in the FST that affects the soundness of the protocol. In addition, our construction enhances security with a similar computation cost as the FST, by requiring two verifications of two different keys, providing a sort of 2-factor authentication. An attacker would need to compromise both the private key s_A corresponding to the public key of A and the shared secret key SK_{AB} to break the protocol.

In our approach, B can verify the honest generation of the challenge using the shared secret key SK_{AB} and the shared counter i_{AB}. This limits an adversary's ability to manipulate the challenge and the proof. Furthermore, updating the shared key SK_{AB} after each use ensures that our protocol remains secure even if the shared secret key is compromised, as the updated key value can be considered a Common Random String (CRS). In the case of a compromised secret key s_A or shared key SK_{AB}, our protocol detects the compromise through a simple verification step during the rejection of an authentication request. This approach is explained in more detail in the following section.

4.3.1 Compromised key detection mechanism

The presence of both keys, namely the secret key s_A and the shared key SK_{AB}, in the proof enables the detection of key compromise, as outlined in Algorithm 2. During the authentication verification of A, the protocol can identify potential key compromise using the following approach: If the challenge c_B was correctly generated using the KMAC function, but the SIS-based proof is invalid, it suggests a possible compromise of the shared key SK_{AB} or an attacker's lucky guess of the challenge. To keep track of such incidents, a counter f_{sym} is employed. If the counter exceeds a certain threshold value, denoted as $f_{threshold}$ and configurable by the verifier, it indicates that the shared secret key SK_{AB} is compromised. Similarly, if the challenge c_B is invalid but the SIS-based proof is valid, the counter f_{asym} is utilized. Upon reaching the threshold $f_{threshold}$, the verifier reports the incident to the trusted authority (TA) using an authenticated request. The trusted authority (TA) must promptly respond and take appropriate action based on the reported issue.

First, if the public/private key pair is compromised, the certificate of the concerned node is revoked, and all the other nodes are informed immediately. Then, a new public/private key pair is generated for the node and certified by the trusted authority (TA) or the corresponding authority. The new keys have to be added manually to the device. Finally, the new

Algorithm 2: Compromised key detection

```
Input  : $c_B$: $B$'s Challenge for $A$, $schnorr_A$: $A$'s Schnorr proof,
         $proof_A$: $A$'s NIZK proof, $SK_{AB}$: Shared secret key,
         $keypair_A$ $A$'s Public/Private key pair, $f_{sym}$: Number of
         wrong challenge authentications, $f_{asym}$: Number of
         wrong Schnorr proofs, $f_{threshold}$: Maximum allowed
         number for $f_{sym}$ or $f_{asym}$
Output: Incident report to Server
// Verify authentication of node $A$
if $proof_A$ is not valid then
    Reject proof;
    if $c_B$ is valid and $schnorr_A$ is not valid then
        // Check if $SK_{AB}$ is compromised
        $f_{sym} = f_{sym} + 1$;
        if $f_{sym} = f_{threshold}$ then
            // $SK_{AB}$ is compromised
            Report incident to $TA$;
    if $schnorr_A$ is valid and $c_B$ is not valid then
        // Check if $keypair_A$ is compromised
        $f_{asym} = f_{asym} + 1$;
        if $f_{asym} = f_{threshold}$ then
            // $keypair_A$ is compromised
            Report incident to $TA$;
    else
        // $SK_{AB}$ and $keypair_A$ are not
           compromised
        exit;
else
    Accept proof;
```

certificate is shared with the other nodes that used to have it. If the shared secret key is compromised, the trusted authority (TA) allows the affected nodes to perform another interactive mutual authentication to establish a new shared key.

5 Evaluation

In this section, we perform an informal (Sect. 5.1) and a formal security analysis (Sect. 5.2) and evaluate the protocol's performance (Sect. 5.3).

5.1 Informal security analysis

Before proving the security of our protocol using the formal security analysis, we will discuss the security of our approach based on an informal analysis for each attack mentioned in Sect. 3.3.

5.1.1 MITM attack

As mentioned in Sect. 3.3, we distinguish between several types of MITM attacks, namely, active attacks, eavesdropping, message modification, and replay attacks.

- Active attacks: According to Dwivedi et al. [21], to protect against active attacks in authentication scenarios, it is crucial to establish a challenge-response method between the prover and the verifier. This method involves the verifier challenging the prover to ensure that their proof is randomized using the verifier's challenge. In our case, we employ a zero-knowledge proof system with an authenticated random challenge through a MAC tag, which implies that our protocol is secure against such attacks.
- Eavesdropping: The security of our protocol against eavesdropping is directly related to the fact that our protocol is zero-knowledge and to the security of the elliptic curve discrete logarithm problem, which is considered to be a hard unbreakable problem as mentioned by multiple sources. Demonstrating that a protocol is zero-knowledge ensures its security against passive attacks like eavesdropping [21].
- Message modification attacks: Our authentication system is secure against such attacks since each message can be authenticated by incorporating it into the MAC function used in the proof, which allows the recipient to detect any malicious message modification and thus ignore the request. In addition, the generated MAC tag is used in the Schnorr proof, where we provide an additional layer of security by verifying if the proof is valid after verifying the MAC tag. However, one may argue that in our protocol setup phase, the MAC function is not utilized. In that case, the purpose of the interaction is to establish a shared secret key between two parties rather than authenticating any message. In other words, if the sent commitment, challenge, or response is altered, the Schnorr proof will not be accepted and thus does not affect the security of our system.
- Replay attacks: Our system is also secure against replay attacks in all their forms. Generating a proof in our non-interactive zero-knowledge protocol involves using a counter that is incremented each time and a new randomized key in each authentication. This way, we ensure that each authentication is unique and secure against replay attacks. We increment the counter twice after each use, once for the challenge and once for the new randomized key, to guarantee that we always get randomized values that can eliminate the risk of a replay attack.

5.1.2 Brute force attack

Our protocol is secure against brute force since we use keys that achieve a 128-bit security level at the lowest. In addition, security against brute force attacks is guaranteed by the completeness and soundness property of our zero-knowledge protocol as mentioned by [21]. Furthermore, we use a combination of a public/private key pair and a shared secret key. The public/private key pair has a length of 256 bits, which means that there are 2^{256} combinations for the secret asymmetric key, while the 256-bit shared secret key also has 2^{256} combinations. This implies that for each one of the 2^{256} possible asymmetric private keys, there exists 2^{256} possible shared secret keys. So, attempting all these combinations is not feasible for the most powerful computers today since the likelihood of succeeding in such an attack is extremely negligible. Even during our one-time setup phase, where we use only the private/public key pair, 2^{256} combinations are still infeasible and that protocol is only used in the first communication.

While this informal security analysis provides us with an overview of the security of our protocol, it is still insufficient. A formal security analysis is necessary to thoroughly test the protocol against these threats. For that reason, we provide such an analysis in the following section.

5.2 Formal security analysis

In this section, the soundness of our protocol will be discussed and proven based on a formal security analysis using Tamarin-prover [40].

5.2.1 Tamarin-prover

Tamarin-prover is a well-known tool used for the automated and unbounded analysis of security protocols against adversaries. It represents the possible executions of a protocol symbolically as a transition system, where each execution corresponds to a specific sequence of events. The state of the transition system is characterized by a multiset of facts, where the initial state is the empty multiset and the following states are mentioned inside the protocol rules. These rules are structured in the following form:

```
rule rule_name:
    l --[ a ]-> r
```

where r denotes the multiset of produced facts, l denotes the multiset of consumed facts that describe the facts that must exist in the multiset before applying the rule, and a denotes the multiset of action facts behaving as a symbolic label. The action facts a_i are used for specifying an event or performing a certain logic, such as accepting a proof. Security properties are defined over them, enabling the analysis of specific security requirements. In addition, consumed facts are used to specify the transition from one rule to another, and they can also contain variables, thereby allowing for more flexibility in rule application. Rules can also be used to specify the capabilities of an adversary, such as their ability to compromise a certain secret key.

Tamarin-prover uses an insecure and untrusted network, as in the Dolev–Yao Adversary model [20]. It also assumes that all the used built-in functions represent perfect cryptography, where decrypting a cipher only means that the adversary knows the required key. In addition to that, custom functions with/or without their corresponding equations can be defined and are considered perfect one-way functions. Furthermore, restrictions can also be used to limit any protocol execution that does not meet the conditions included in it. Finally, lemmas can be defined using the keyword "lemma" to specify a proposition or a statement that has to be proved or established. The lemmas in Tamarin-prover are formulated with action traces over a specified timeline. The Tamarin-prover will attempt to prove or disprove the lemma based on the specified rules and constraints defined in the Tamarin model. In case of disproving the lemma, a counter-example will be provided.

5.2.2 Modeling our protocol in tamarin

Modeling our protocol posed a challenge because Tamarin-prover does not provide built-in functions or examples that help with modeling the Zero-Knowledge protocol and protocols that are not based on primitive cryptographic functions. The main challenge we encountered was modeling the proof generation and verification process, specifically with respect to the logic of Schnorr's proof. We found limited examples or studies that specifically addressed the modeling of Zero-Knowledge Proofs. While references such as [2–4] acknowledged this challenge and claimed to provide solutions, we were unable to access their examples written in Tamarin code due to expired or nonexistent links. Furthermore, Fischlin [25] conducted a study on the works of [2–4], providing several examples of modeling a general non-interactive zero-knowledge proof system using Tamarin-prover. However, it should be noted that the provided examples differ from our case as he utilizes the built-in encryption functions of Tamarin in his proofs. Therefore, his model would require modifications to adapt to our specific scenario. One of the initial suggestions proposed in [25] was to accomplish proof generation and verification through predefined functions. Initially, we attempted to implement two functions for our modeling efforts: one for generating a proof and another for verifying a proof generated using the former function. However, we encountered significant challenges when it came to representing the equations required for generating and verifying the Schnorr elliptic curve proof.

Defining simple abstract equations, as commonly done in most examples, leads to multiple compiler errors and warnings in Tamarin. This is because Tamarin does not permit the use of built-in functions and public values, such as elliptic curve multiplication and the public generator point, within functions, equations, and restrictions. To overcome this limitation, we attempted to define a custom elliptic curve multiplication function, as demonstrated in [42]. However, this approach does not adhere to the properties of elliptic curves. Instead, it represents it symbolically and considers it as a one-way function, which is enough regarding that the goal of using the elliptic curve multiplication is to make the secret key not extractable from the output of the elliptic curve multiplication. For modeling the proof verification, we chose to utilize restrictions, as described in [25]. This approach excludes protocol traces with non-valid proof runs.

Despite using restrictions instead of equations, Tamarin-prover continued to generate warning messages regarding the validity of the restrictions due to its built-in elliptic curve multiplication functionality. These warnings indicated that the analysis results might be incorrect. To address this, we defined our elliptic curve multiplication function. To ensure the validity of our model, we imposed a restriction on the function's input, treating the constant generator point P as an internal constant of the procedure. The formalization of this restriction is as follows:

```
restriction IsValidZkp:
 "
   All #i w r cl cr pkey R.
   ValidSchnorrZKP(schnorrZKProof(w,
       r, cl), pkey, R, cr)@i
     ==> (Ex m n. pkey = multp(m) &
        R = multp(n) &
     m = w & n = r & cl = cr)
 "
```

Whenever the action `ValidSchnorrZKP` is called, this restriction `IsValidZkp` ensures that the correct elements are provided for verifying the proof `schnorrZKProof`. It checks if the verifier has entered the same challenge used in the proof generation (`cl = cr`), the appropriate public key `pkey` corresponding to the EC multiplication (`multp`) of the private key `w` used in generating the proof, `pkey = multp(m) & m=w`, and the proper commitment `R` corresponding to the elliptic curve multiplication of the random value `r` used in the proof generation. We adopted this concept because we did not find any examples of modeling non-primitive proof generation and verification functions.

Furthermore, our protocol includes the Schnorr proof and a MAC tag for generating and verifying the challenge. We modeled the MAC function using the classical approach as an equation for verifying the MAC is sufficient. The MAC tag is verified prior to verifying the Schnorr proof. If the tag is incorrect, the verification fails; otherwise, the Schnorr proof is verified to determine the acceptance of the proof. We created rules for various protocol steps, including generating the necessary keys and counters, managing key disclosure to adversaries, initializing participants in the communication, sending and verifying proofs, and calculating shared keys.

5.2.3 Lemmas construction

The Tamarin model is incomplete without specifying the lemmas that help Tamarin-prover to investigate the required security properties. For each Tamarin model of our protocols, we provided a lemma that proves that our model is executable. Tamarin will only try to find at least a single protocol trace, where the provided model is executable with the expected order between two distinct participants and the secret key of the prover and the shared secret key are not compromised. We have also provided several lemmas for proving the properties of ZKP (i.e., completeness, soundness, and zero-knowledge) and the security of our model. We based our lemmas' representation on the examples provided by [25] with several changes to adapt it to our model. We have also added several other lemmas concerning the properties of an authentication system. Particularly, we proved the following lemmas:

Lemma 1 *Completeness: The completeness property of our NIZK transformation is defined as follows:*

In this lemma, we prove that every time a verifier receives valid NIZK proofs, the verifier can verify them as correct. In other words, we say that for every verifier `B` with ID = `id`, private key (witness) `w`, random initial number `r`, challenge `c`, public key `pkey`, and commitment `R`, where `pkey` is the EC multiplication of `w` (`pkey = multp(m) & m = w`), `R` is the EC multiplication of `r`, `B` received the proof `schnorrZKProof` of the prover via the action `ReceivedZkp`, and the protocol has finished running, then, `B` has verified the proof as valid (`VerifiedSchnorrZKP`).

```
lemma completeness:
"
    All B id w r c pkey R #i1 #i3.
    ( Ex m n. pkey = multp(m) & R =
       multp(n) & m = w & n = r ) &
    ReceivedZkp(B, id, schnorrZKProof
       (w, r, c), pkey, R, c)@i1 &
    Finish(B, id)@i3
    ==> ( Ex #j.
    VerifiedSchnorrZKP(B, id,
       schnorrZKProof(w, r, c), pkey
       , R, c)@j )
"
```

Lemma 2 *Soundness: For the soundness property, the following lemma is constructed:*

In this lemma, we state that if a proof `schnorrZKProof` is verified correctly (`VerifiedSchnorrZKP`), then it is generated with the righteousness elements, and the secret key is not revealed to the verifier, or that the adversary did not compromise it using any other way (`KU(w)`). In this case, we utilized the or constructor "|", meaning that either the *prover* `A` *knows the witness* `w` *(*`KnowsWitness (A, w )`*) and generates a valid proof where the secret key is not revealed through* `RevealSecret(A)` *or the proof is not correct, and the witness is still not revealed.*

```
lemma soundness:
"
    All B id #i1 w r cl cr pkey R.
    VerifiedSchnorrZKP(B, id,
        schnorrZKProof(w, r, cl),
        pkey, R, cr)@i1
    ==> ( Ex m n.
    pkey = multp(m) & R = multp(n) &
        m = w & n = r & cl = cr ) &
        (
            ( Ex A #i. KnowsWitness(A
                , w)@i)
            | ( Ex A #i. RevealSecret
                (A)@i )
            | ( Ex #i. KU(w)@i )
        )
"
```

Lemma 3 *Zero-Knowledge: For the zero-knowledge property, the following lemma is utilized:*

In this lemma, we state that for every created proof following our protocol, no intruder `K` *or verifier can learn anything about the witness* `w` *(secret key) except when it's revealed by the prover* `A` *through* `RevealSecret(A)`.

```
lemma zero_knowledge:
"
    not(
        Ex A w #i2 #i3.
        CreatedSchnorrZKPForWitness(A
            , w)@i2
        & K(w)@i3
        & not (Ex #j. RevealSecret(A)
            @j)
    )
"
```

Lemma 4 *Message Integrity: Additionally, to the basic properties of a ZKP, we also proved resilience to message modification attacks. To prove this lemma, we added a rule where the adversary can intercept messages from* `A` *to* `B`*, before reaching* `B`*, modify their content, and forward the modified message to* `B`*. In this case, we make sure that if* `B` *accepted the proof as valid, then this implies that the adversary did not change the original message.*

Lemma 5 *Resilience to Replay Attacks: For this lemma, we demonstrated that any attempt to reuse previously validated proofs as valid proofs will lead to their rejection. This outcome aligns with our protocol's requirement for the use of incremented counters and new random values in each new proof.*

5.2.4 Analysis findings

During the formal security analysis of our three sub-protocols using Tamarin-prover, the verification process was successfully completed without encountering any issues. All the lemmas were examined and verified, ensuring the protocol's completeness, soundness, zero-knowledge property, and resilience against various attacks. The results of the lemma verification can be examined and inspected in detail here.[4]

5.3 Performance analysis

5.3.1 Testbed

To implement and test the protocol, we chose to use the programming language Rust, which is known for having performance similar to C and C++ but brings more security compared to them, especially for memory-related attacks. Rust also provides cross-platform compilation, allowing us to develop and test our prototype for different Hardware classes, including ARMv6 and ARMv7 processors. Our protocol was implemented in the form of a Rust crate (library) based on several other cryptography crates. We utilized the crate `linux-keyutils`[5] crate for managing secret keys. The crate`curve25519-dalek`[6] was employed for elliptic curve cryptography since it is designed for secure and high-speed performance, achieving record-breaking cycle counts on various processors compared to other common curves [7]. Finally, the SHA-3 and KMAC functions were operated using the `tiny-keccak`[7] crate, which offers a stable and fast implementation of both functions. The complete implementation of our protocol, along with examples and the security analysis, has been made open source.[8] We have used two Raspberry Pi devices to conduct our experiments. The characteristics of these devices are summarized in Table 1, and no crypto-accelerators were employed. The devices were interconnected wirelessly using a router AVM Fritz!Box 6660 Cable. We positioned both devices at a distance of approximately 10 ms from each other.

5.3.2 Experiments

We conducted several experiments to test the performance of our protocol on the two hardware platforms mentioned in Table 1. In the first experiment, we measure the required time

[4] Link: https://github.com/tum-esi/act-nizkp/tree/main/tamarin_security_analysis.

[5] Link: https://lib.rs/crates/linux-keyutils.

[6] Link: https://lib.rs/crates/curve25519-dalek.

[7] Link: https://lib.rs/crates/tiny-keccak.

[8] Link: https://github.com/tum-esi/act-nizkp.

Table 1 Hardware platforms used in the measurement and their properties

Devices	CPU	RAM	OS
Raspberry Pi 3 Model B V1.2	Quad-Core 1.2GHz Broadcom BCM2837 64B	1GB	Raspberry Pi OS Lite 32B
Raspberry Pi Zero W V1.1	Single-Core 1GHz Broadcom BCM2835 32-bit	512MB	Raspberry Pi OS Lite 32B

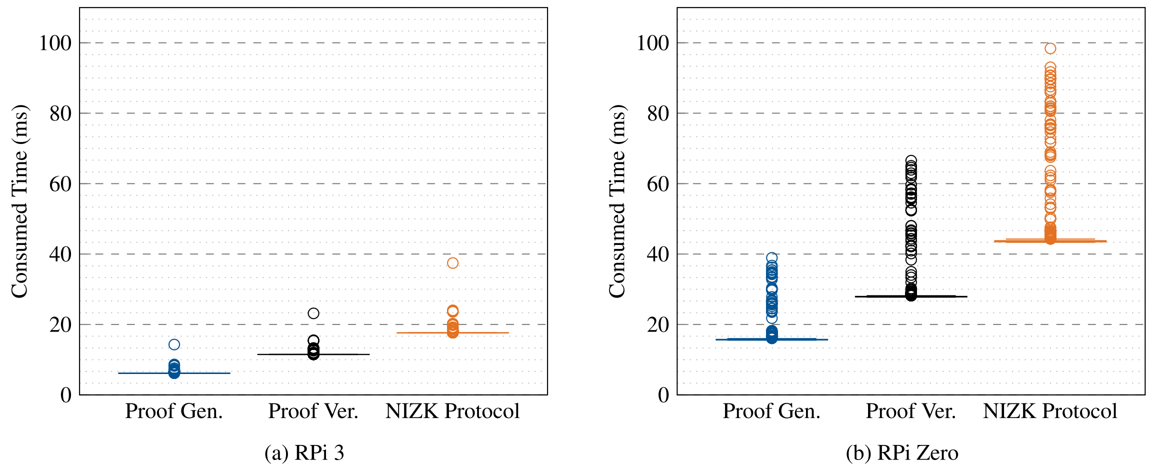

Fig. 3 Execution time for proof generation (proof Gen.), proof verification (proof Ver.), and NIZK protocol on **a** Raspberry Pi 3 and **b** raspberry Pi Zero W

to perform the NIZK proof generation and the NIZK proof verification on each hardware platform (both the verifier and the prover run on the same device). We have repeated the measurement 1000 times. The result of this measurement is shown in Fig. 3. In the figure, we observe that the interquartile range (IQR), which represents the range between the 25th and 75th percentiles of the data, along with the whiskers (representing values within 1.5 times the IQR), and the median values are all closely clustered together. This indicates a consistent performance pattern of our NIZK protocol on the same device. Also, we notice that the proof verification is slower than the proof generation. This was expected since the proof generation requires one EC multiplication (see Eq. 16), while the proof verification requires two EC multiplications (see Eq. 19). Finally, we can observe that the NIZK protocol is relatively faster on the Raspberry Pi 3 (see Fig. 3a) compared to the case when it runs on Raspberry Pi Zero W (see Fig. 3b).

In the second experiment, we measured the execution time of our proposed protocol, including the setup phase and the NIZK transformation, when the verifier and the prover were running on different devices. We ran the prover on the Raspberry Pi Zero W as a TCP client and the verifier on the Raspberry Pi 3 as a TCP server. The reason for this selection is that, in most IoT scenarios, the verifier role is typically assigned to the more powerful device. We started the different protocol phases from the client side as the prover, and we measured the time from the first request until receiving

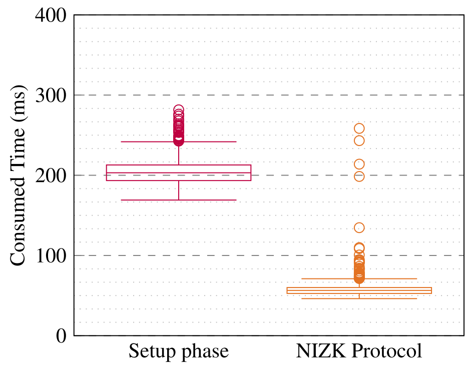

Fig. 4 Time consumed by the setup phase and the NIZK protocol

the last response from the verifier. We repeated each measurement for 1000 times. We added a verifier response to the NIZK protocol to confirm that the verifier has completed the verification of the proof. This will help the prover determine when to stop measuring the execution time.

The results are illustrated in Fig. 4, which shows that the setup phase has slightly larger variability in performance compared to the NIZK protocol. This behavior is expected due to the interactive nature of the setup phase. However, this is not a problem in our case since the setup phase will be used *only once* during the initial communication between two devices to securely share a secret key or to reset the counter.

Finally, we conducted the same setup to compare our NIZK protocol with the 3-round SIS. We have also added a verifier response in the SIS protocol to inform the prover

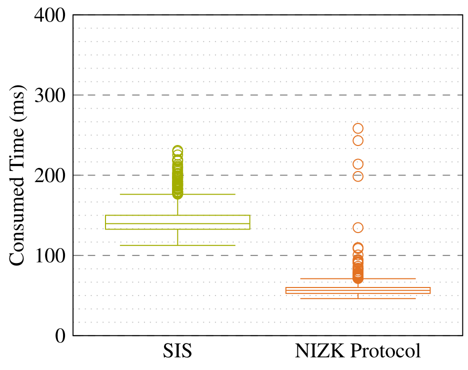

Fig. 5 Execution time comparison between the SIS protocol and the NIZK protocol

when the verification of the proof is complete. This enables the prover to accurately measure the execution time by knowing when to stop the timing measurement. We performed the measurement for each protocol a total of 1000 times. It is important to note that the measurement conducted for our NIZK protocol did not include the setup phase. Figure 5 presents the execution time of this experiment for both protocols. The results indicate that our NIZK transformation achieves a significant reduction in execution time, with the protocol requiring approximately 50% less time for proof generation and verification compared to the original SIS protocol. Furthermore, our protocol incorporates two-factor verification, enhancing overall security. In addition, we notice that the SIS has larger variability in performance compared to our NIZK protocol because of the more spread-out distribution it shows. This behavior is expected in the SIS, given its interactive nature that contains more interactions between the prover and the verifier required to complete the proof verification.

In conclusion, the experiments conducted to evaluate the performance of our NIZK protocol on different hardware platforms and in various scenarios have provided valuable insights validating the performance and effectiveness of our proposed solution. The experiments showed that our NIZK protocol improves upon the performance of traditional approaches (e.g., SIS and ECDHE) while leveraging more security through two-factor authentication.

6 Related work

This section provides an overview of existing research and studies relevant to our proposed protocol. Furthermore, we compare our proposed solution with other existing works that aimed to improve the soundness of FST (see Sect. 6.1).

Many works focused on studying the soundness of the required hash function of FST, aiming to provide a more secure procedure. For instance, many proposed utilizing correlation intractable hash functions to make the FST sound as done by Canetti et al. [10–12] and Holmgren et al. [29]. However, these solutions do not apply to all types of utilized cryptographic primitives. Mittelbach et al. [36] presented a hash function that guarantees the security of the FST for a specific group of protocols, called “highly sound” protocols, that have not been studied before. These highly sound protocols are ZKP that satisfy three additional properties: the honest prover computes the commitment independently of the witness and the instance being proven, the protocol has a significantly small soundness error, and honest conversations between the prover and verifier can be simulated knowing just the input. This hash function is called a q-wise independent hash function. It is a type of hash function that is defined to map inputs from a set I to outputs in a set O, where the corresponding q outputs in O are uniformly distributed and independent from each other for any distinct q elements in I. This property ensures a high level of randomness and unpredictability. However, The proposed solution is restricted to a specific group of Σ-protocols and hash functions, which limits its applicability.

Using a traditional hash function was not the only proposed approach since several other solutions were also investigated and considered. For example, Lindell [34] employed a Common Reference String (CRS) and a dual-mode commitment scheme with a Non-programmable random oracle(NPRO)[9] instead of the classic hash function to perform the transformation. The soundness of their method relies on NPRO, which is a better method than the normal random oracle. However, this approach is restricted to a specific group of Σ-Protocols. Similarly, Ciampi et al. [16] proposed a new transformation, also based on NPRO, as efficient and as general as the FST improving upon the method of Lindell [34]. The proposed approach was based on weaker requirements as in Lindell’s approach and has also improved efficiency over it. Even though it is based on a dual commitment scheme, it is still not as secure as our method. Moreover, Bellare et al. [5] presented a security proof of the FST in certain cases based on the standard model instead of the random oracle model. They used it in converting three-move authentication protocols into two-tier signature schemes with the condition that the starting protocol must have security against concurrent attacks, meaning that it restricts the allowed Σ-protocols. A two-tier scheme involves a prover who possesses a primary public key and a corresponding primary secret key to prove his identity through a signature. At every new signature, the prover generates a new set of secondary public and secret keys and uses these in combination with the primary keys and the message to produce the signature. Verification of the

[9] The non-programmable random oracle (NPRO) is a type of random oracle that functions as a black box, but it cannot be manipulated or programmed by a simulator or the adversary.

signature necessitates the use of both the primary and the secondary public keys linked to the message, which makes their approach requires transmitting 32B of additional data in each communication. Chen et al. [14] modified the FST to create a non-interactive deniable authentication protocol that preserves the deniability property of cryptographic systems. The main difference between the modified scheme and the original one is the replacement of the one-way hash function with elliptic curve cryptography (ECC)-based ElGamal encryption to produce a random challenge. However, the proposed solution not only restricts the permissible group of the cryptographic primitive used but also necessitates the transmission of 160B in each request.

Other methods based on homomorphic hash and encryption functions were also proposed by several authors. Maurer et al. [35] presented a new general group of certain Σ-protocols that describes them as a single general protocol using one-way homomorphic hash functions, since, according to the authors, Σ-protocols are similar in their construction and can be proven the same when considered in the right abstraction level. However, their approach restricts the allowed hash functions and Σ-protocols to a specific group. Similarly, Iovino et al. [30] proposed a transformation function for transforming 3-round ZKP that uses relations related to quadratic residuosity and RSA into non-interactive ones, based on using the group of homomorphic one-way functions as in [35]. They improved upon the work of [35] by adding more requirements to that group of functions, such as the existence of a trapdoor, creating a new group called special one-way homomorphic functions. Yet, their solution limits the allowed hash functions and Σ-protocols as in [35]. Damgard et al. [18] limited the group of cryptographic primitives and used homomorphic encryption[10] instead of homomorphic hash functions to convert any 3-round ZKP protocol, Σ-protocols, into NIZK protocols. Chaidos et al. [13] provided an instantiation of [18] using homophobic encryption that does not require a random oracle model, making the required assumptions for its security less complicated. However, their technique is still limited to just the group of homomorphic encryption.

While previous research has primarily focused on improving the security of the Fiat–Shamir transformation (FST), a significant gap exists in the literature. Most of these approaches concentrated solely on enhancing the FST's security without considering the original Σ-protocol's security. Some imposed limitations by restricting the choice of cryptographic primitives or Σ-protocols. Notably, no prior research explored the possibility of creating a NIZK transformation that not only addresses FST security concerns but also enhances the original Σ-protocol's security without imposing restrictions on cryptographic primitives or Σ-protocols. This research aims to bridge these gaps, providing a comprehensive, efficient, and robust solution for secure IoT device authentication.

[10] Homomorphic encryption is a form of encryption that permits calculations to be conducted on encrypted data, without needing to decrypt it in the first place. This means that the computations can be carried out on the encrypted data, preserving its confidentiality. The result of the computations can be decrypted to obtain the result of the computations.

6.1 Comparative analysis

We compare our work with existing research focusing on the following aspects: **P1**: the adopted cryptographic primitives **P2**: the solution validity for all types of the adopted cryptographic primitive **P3**: the solution validity for all Σ-protocols **P4**: the number of commitments **P5**: The improvement on the security of the Σ-protocols **P6**: the data size that needs to be saved in prover's memory **P7**: the data size to be saved in verifiers' memory **P8**: the data size to be transmitted for the proof. For all memory-related comparisons, we assume that ECC is utilized with any keys, commitments, challenges, or responses being 32 bytes values.

While previous solutions have mainly centered around utilizing specific groups of a particular cryptographic primitive (e.g., [10–14, 18, 30, 35, 36]) or Σ-protocols (e.g., [5, 12, 30, 34–36]), as depicted in Table 2, our research takes a different approach. Instead of using a hash or an encryption function, we chose to employ a MAC and introduced a novel concept known as the "authenticated-challenge transformation". Our approach does not only present a non-interactive proof system derived from any Σ-protocol but also incorporates an integrated non-interactive challenge verification. As a result, the verifier gains the ability to verify the honest generation of the challenge using any group of MAC functions. Furthermore, our approach enhances security compared to other existing solutions by necessitating two secret keys in the proof, which adds a layer of complication for potential compromises. Additionally, we ensured the randomness of the challenge by updating the shared secret key and shared counter after each use, which makes sure that the challenge is always random, even if the shared secret key is compromised. Compromising the shared secret key does not impact the soundness of our proof, as the adversary is still forced to replace it with an unpredicted random value after each use which results in a random challenge each time, similar to the approaches based on CRS (e.g., [12, 16, 34, 36]). This aspect strengthens the security of our transformation relative to the traditional FST.

Even though Ciampi et al. [16] solution does deliver validity for all types of the employed cryptographic primitive and Σ-protocols with improved security throughout the use of dual commitments, our approach still provides better security improvements through the use of two-factor authentication in the proof. Dual commitment-based approaches (e.g., [16,

Table 2 Comparison of our proposed solution with related work

Related work	Year	P1	P2	P3	P4	P5	P6	P7	P8
Canetti et al. [12]	2019	Hash	○	○	1	○	64B + CRS	32B + CRS	96B
Canetti et al. [10]	2016	Hash	○	●	1	○	64B	32B	96B
Canetti et al. [11]	2018	Symmetric Encryption	○	●	1	○	64B	32B	128B
Holmgren et al. [29]	2018	Hash	○	●	1	○	64B	32B	96B
Mittelbach et al. [36]	2018	Hash	○	○	1	○	64B + CRS	32B + CRS	96B
Lindell et al. [34]	2014	NPRO	●	○	2	◑	64B + CRS	32B + CRS	96B
Ciampi et al. [16]	2016	NPRO	●	●	2	◑	64B + CRS	32B + CRS	96B
Bellare et al. [5]	2007	Hash	●	○	1	●	64B	32B	128B
Chen et al. [14]	2010	Asymmetric Encryption	○	●	1	●	96B	96B	160B
Maurer [35]	2015	Hash	○	○	1	○	64B	32B	96B
Iovino et al. [30]	2019	Hash	○	○	1	○	64B	32B	96B
Damgård et al. [18]	2006	Asymmetric Encryption	○	●	1	○	96B	96B	96B
Chaidos et al. [13]	2015	Asymmetric Encryption	○	●	1	○	96B	96B	96B
Our approach	2023	MAC	●	●	1	●	100B	68B	96B

P1: Cryptographic Primitives (Hash, asymmetric or asymmetric encryption), **P2**: Validity for all cryptographic primitives (Yes: ●, No: ○)
P3: Validity for all Σ-protocols (Yes: ●, No: ○), **P4**: Number of Commitments
P5: Improving the security of the Σ-protocols(Yes: ●, Partially: harveyBallHalf, No: ○), **P6**: Prover Memory Data Size
P7: Versifier Memory Data Size, **P8**: Size of transmitted Proof

34]) do make it harder to cheat in the FST. However, it does not affect the security of the original Σ-protocol. It is worth noting that our protocol requires slightly more storage compared to certain other methods (e.g., [5, 10, 11, 29, 30, 35]). However, this increase in storage comes as a trade-off for the enhanced level of security it offers. Also, We were unable to calculate the exact required storage for protocols such as [12, 16, 34, 36] as no information about the size of the was mentioned. Nevertheless, considering that the CRS should not be smaller than 32B when using ECC, as mentioned earlier, implies that these protocols may require larger storage than ours.

7 Conclusion

There is a growing demand for secure authentication mechanisms in IoT systems. In this work, we propose a solution that overcomes the limitations of traditional authentication mechanisms by introducing a NIZK protocol that enables secure and private communications with two-factor authentication. Specifically, the work focuses on transforming interactive 3-round Σ-protocols into NIZK protocols based on the FST. The proposed solution enhances the soundness of the FST and the original Σ-protocol by creating a non-interactive proof scheme that requires two verifications with two distinct keys, improving the overall robustness and soundness of the authentication process. The security of the proposed solution was formally proved using Tamarin-Prover. In addition, we provided a practical implementation of the proposed protocol in the Rust programming language and conducted a performance analysis. The results clearly demonstrated that our NIZK protocol outperforms the SIS protocol by requiring nearly 50% less time for execution.

Our approach focuses on enhancing the security of the Fiat–Shamir transformation (FST) for IoT authentication scenarios while also improving upon the security of the original Σ-protocol. The primary improvement lies in the introduction of a mechanism to verify that the prover has generated the verifier's challenge without resorting to dishonest methods. This effectively mitigates the risk of fraudulent challenge creation and manipulation of the proof. This enhancement is achieved by incorporating a MAC tag, which includes an incremented counter and a randomized shared secret key, replacing the use of a hash function. This change also transforms the solution into a two-factor authentication system, relying on two distinct keys for added security. As demonstrated in our performance analysis, our transformation does not compromise the protocol's efficiency; in fact, it enhances the original Σ-protocol's efficiency while introducing an additional layer of security through two-factor verification. Furthermore, in our comparative analysis, we have shown that no other proposed solution matches the level of security we offer. We have also illustrated that despite this heightened security, our protocol still transmits approximately the same amount or even less data compared to other proposed approaches. Another significant improvement in our approach is its flexibility. Unlike many other solutions (as discussed in Sect. 6.1), we do not restrict the use of specific groups of cryptographic primitives or Σ-protocols. Our transformation can be applied to any 3-round ZKP (Σ-protocol) utilizing any secure MAC function, as our chosen cryptographic primitive is a MAC function.

Although the proposed NIZK authentication protocol provides wide applicability in the domain of IoT, further research can focus on optimizing its scalability to handle multi-party authentication and session key generation since our approach is still limited to authentication between two devices.

Author contributions FH did protocol proposition, writing original draft, formal security analysis, and software development. MH did validation, visualization, writing-review and editing, and supervision. DS provided resources, did review, and supervision. SS did review, editing, and supervision.

Funding Open Access funding enabled and organized by Projekt DEAL. The authors did not receive support from any organization for the submitted work.

Data availability The datasets utilized during the experiments and the code developed to implement the proposed solution are publicly available on the institute's GitHub page. You can find the relevant data here: https://github.com/tum-esi/act-nizkp/.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical standards This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Abi-Char, P.E., Mhamed, A., Bachar, E.-H.: A fast and secure elliptic curve based authenticated key agreement protocol for low power mobile communications, In: The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2007), pp. 235–240. IEEE (2007)
2. Backes, M., Unruh, D.: Computational soundness of symbolic zero-knowledge proofs against active attackers. In: 2008 21st IEEE Computer Security Foundations Symposium, pp. 255–269. IEEE (2008)
3. Backes, M., Bendun, F., Unruh, D.: Computational soundness of symbolic zero-knowledge proofs: weaker assumptions and mechanized verification. POST **13**, 206–225 (2013)
4. Backes, M., Maffei, M., Unruh, D.: Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In: 2008 IEEE Symposium on Security and Privacy (SP 2008), pp. 202–215. IEEE (2008)
5. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In: Public Key Cryptography–PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography Beijing, China, April 16–20, 2007. Proceedings 10, pp. 201–216. Springer (2007)
6. Bernhard, D., Pereira,O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2–6, 2012. Proceedings 18, pp. 626–643. Springer (2012)
7. Bernstein, D.J.: Curve25519: new Diffie–Hellman speed records. In: Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24–26, Proceedings 9, pp. 207–228. Springer (2006)
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26–30, 2013. Proceedings 32, pp. 313–314. Springer (2013)
9. Bitansky, N., Dachman-Soled, D., Garg, S., Jain, A., Kalai, Y.T., López-Alt, A., Wichs, D.: Why" Fiat–Shamir for Proofs" Lacks a Proof. In: TCC, vol. 7785, pp. 182–201. Springer (2013)
10. Canetti, R., Chen, Y., Reyzin, L.: On the correlation intractability of obfuscated pseudorandom functions. In: Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10–13, 2016, Proceedings, Part I 13, pp. 389–415. Springer (2016)
11. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat–Shamir and correlation intractability from strong KDM-secure encryption. In: Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29–May 3, 2018 Proceedings, Part I 37, pp. 91–122. Springer (2018)
12. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat–Shamir: from practice to theory. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, pp. 1082–1090 (2019)
13. Chaidos, P., Groth, J.: Making sigma-protocols non-interactive without random oracles. In: Public-Key Cryptography–PKC 2015: 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30–April 1, 2015, Proceedings, pp. 650–670. Springer (2015)
14. Chen, Y., Chou, J.-S., Lin, C.-F.: A novel non-interactive deniable authentication protocol with designated verifier on elliptic curve cryptosystem. Cryptology ePrint Archive (2010)
15. Chen, Z., Jiang, Y., Song, X., Chen, L.: A survey on zero-knowledge authentication for internet of things. Electronics **12**(5), 1145 (2023)
16. Ciampi, M., Persiano, G., Siniscalchi, L., Visconti, I.: A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. In: Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10–13, 2016, Proceedings, Part II 13, pp. 83–111. Springer (2016)
17. Damgård, I.: On σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science, 84 (2002)
18. Damgård, I., Fazio, N., Nicolosi, A.: Non-interactive zero-knowledge from homomorphic encryption. In: Theory of Cryptography: 3rd Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4–7, 2006. Proceedings 3, pp. 41–59. Springer (2006)

19. Dammak, M., Boudia, O.R.M., Messous, M.A., Senouci, S.M., Gransart, C.: Token-based lightweight authentication to secure IoT networks. In: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1–4. IEEE (2019)
20. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
21. Dwivedi, A.D., Singh, R., Ghosh, U., Mukkamala, R.R., Tolba, A., Said, O.: Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for internet of things. J. Ambient Intell. Human. Comput. 1–11 (2021)
22. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.: Magic functions: in memoriam: Bernard m. dwork 1923–1998. JACM **50**(6), 852–921 (2003)
23. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat–Shamir transform. In: INDOCRYPT, vol. 7668, pp. 60–79. Springer (2012)
24. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Crypto, vol. 86, pp. 186–194. Springer (1986)
25. Fischlin, S.: Formalising zero-knowledge proofs in the symbolic model. Master's thesis, ETH Zurich (2021)
26. Goldwasser, S., Kalai, Y.T.: On the (In)security of the Fiat–Shamir paradigm. In: 44th Annual IEEE Symposium on Foundations of Computer Science. Proceedings., 2003, pp. 102–113 (2003). https://doi.org/10.1109/SFCS.2003.1238185
27. Haller, N.: The S/KEY one-time password system. Technical report (1995)
28. Hao, F.: Schnorr non-interactive zero-knowledge proof. Technical report (2017)
29. Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 850–858. IEEE (2018)
30. Iovino, V., Visconti, I.: Non-interactive zero knowledge proofs in the random oracle model. In: Codes, Cryptology and Information Security: 3rd International Conference, C2SI 2019, Rabat, Morocco, April 22–24, 2019, Proceedings-In Honor of Said El Hajji, pp. 118–141. Springer (2019)
31. Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat–Shamir for proofs. In: Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37, pp. 224–251. Springer (2017)
32. Kelsey, J., Chang, S.-J., Perlner, R.: Sha-3 derived functions: cshake, kmac, tuplehash, and parallelhash. NIST Spec. Publ. **800**, 185 (2016)
33. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
34. Lindell, Y.: An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. Cryptology ePrint Archive (2014)
35. Maurer, U.: Zero-knowledge proofs of knowledge for group homomorphisms. Des. Codes Crypt. **77**, 663–676 (2015)
36. Mittelbach, A., Venturi, D.: Fiat–Shamir for highly sound protocols is instantiable. Theoret. Comput. Sci. **740**, 28–62 (2018)
37. Mumtaz, M., Akram, J., Ping, L.: An RSA based authentication system for smart IoT environment. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 758–765 (2019)
38. Nyangaresi, V.O., Ogundoyin, S.O.: Certificate based authentication scheme for smart homes. In: 2021 3rd Global Power, Energy and Communication Conference (GPECOM), pp. 202–207. IEEE (2021)
39. Santoso, F.K., Vun, N.C.: Securing IoT for smart home system. In: 2015 International Symposium on Consumer Electronics (ISCE), pp. 1–2. IEEE (2015)
40. Schmidt, B., Meier, S., Cremers, C., Basin, D.: Automated analysis of Diffie–Hellman protocols and advanced security properties. In: 2012 IEEE 25th Computer Security Foundations Symposium, pp. 78–94. IEEE (2012)
41. Stinson, D.R., Paterson, M.B.: Cryptography: Theory and Practice. Chapman and Hall/CRC, Boca Raton (2018)
42. Whitefield, J.D.: Formal analysis and applications of direct anonymous attestation. PhD thesis, University of Surrey (2020)
43. Wu, H., Wang, F., et al.: A survey of noninteractive zero knowledge proof system and its applications. Sci. World J. **2014** (2014)