Password-Based Authentication with Zero-Knowledge Proof of Quadratic Residuosity

Jakob Povsic

2020

Contents

Abstract

We design an authentication protocol that can be used to authenticate users over a network with a username and password. The protocol uses the zero-knowledge proof (ZKP) of quadratic residuosity protocol as a verification mechanism. It is designed on top of the Extensible Authentication Protocol (EAP) framework as an EAP method. The ZKP verification protocol yields interesting security properties that make the protocol favourable to be used over insecure networks.

1 Introduction

Authentication is a core component of computer security and an indispensable part of our modern digital lives. In this thesis we design an authentication protocol using Zero-Knowledge Proofs (ZKPs), an interesting cryptographic phenomenon that reveal nothing more than their validity of its proofs. Our protocol enables network authentication using a username and password. To create a secure password authentication protocol we have to be aware of the common pitfalls and how modern security systems handle them. We use the Extensible Authentication Protocol (EAP) as the framework on top of which we design our authentication protocol.

2 Authentication

Authentication is the process of proving a claim or an assertion. Today the most common application of authentication is in information security systems, however the methods of authentication are not limited to computer science and are also used in fields of archeology, anthropology and others.

In information systems authentication is used for establishing access between restricted system resources and users through digital identities. Government and international institutions have developed guidelines for managing digital identities and authentication processes.

We focus on password based authentication, its security assumptions and tools used in practical applications for increasing security and minimising damage in security breaches.

2.1 Authentication in Information Security

Authentications is a method used in information security to manage access between restricted system resources and an external user wishing to access them.

As defined in RFC-4949 [38], authentication is ”the process of verifying a claim that a system entity or system resource has a certain attribute value.” This is a broad definition, and it most frequently applies to the verification of users identity (e.g at login), however assertions can be made and verified about any subject or object. The process of authentication is done in two parts, *identification* and *verification*.

Identification Presenting an identifier to the authentication system, that establishes the entity being authenticated. In common user authentication systems this is a username or an email verified in the registration process. The identifier needs to be unique for the entity it identifies. The identity of the subject/object can also be pre-determined, so this process is not necessarily visible to the user.

Verification Presenting or generating authentication information that can be used to verify the claim. Commonly used authentication information are passwords, one-time tokens, digital signatures.

2.2 The NITS Model for Digital Identity

Digital Identity Guidelines [25] published by the National Institute of Standards and Technology (NIST) describes a simple digital identity model, that provides a generic authentication framework.

The process has distinct steps of *Enrolment* and *Authentication*.

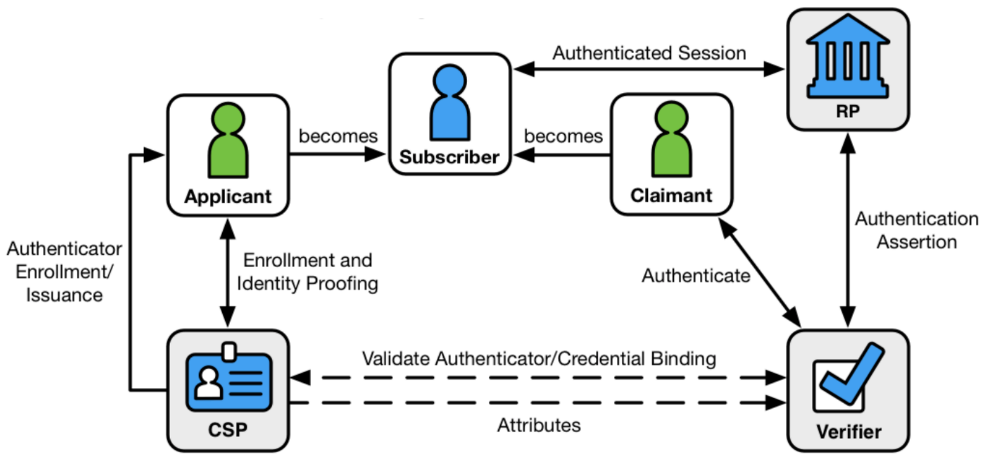

Figure 1: NITS Digital Identity Model

Enrolment The enrolment is a process where an *applicant* becomes a *subscriber* after being successfully *proofed* by a *CSP*. The subscriber is issued a *credential* and one or more *authenticators*.

A common application of this process is *user registration* on websites.

Authentication The *claimant* begins authentication with the *v*erifier by sharing the credential and the authenticators. The *v*erifier validates binding between the credential and authenticators with the *C*SP. An authenticated connection is established between the *s*ubscriber and the *R*P after and assertion is provided by the *C*SP or the *v*erifier to the *R*P.

A common application of this is *user login* on websites.

Note on delegation of roles In the digital identity model 1 roles of *C*SP, *v*erifier and *R*P are distinct in their responsibility. In practice however all these roles can be performed by a single party (e.g any website with native registration and login).

In OAuth2's [26] authentication layer, the resource owner has the roles of *a*pplicant, *c*laimant and *s*ubscriber. The authorisation server has the roles of *C*SP and *v*erifier. The OAuth2 client has the role of the *R*P.

2.3 Authentication Factors

As described in [18] authentication systems can rely on three distinct "factors".

- **Knowledge factors** - Something the user **knows** (e.g, password, security question, PIN)
- **Ownership factors** - Something the user **owns** (e.g, ID card, security tokens, mobile devices)
- **Inherence factors** - Something the user **is** or **does** (e.g, static biometrics - fingerprints, retina, face. dynamic biometrics - voice patterns, typing rhythm)

Strong authentication As defined by governments and financial institutions [36, 19], secure authentication is based on two or more authentication factors. This is also referred to as *multi-factor authentication*.

2.4 Password Based Authentication

Passwords are one of the most common and oldest forms of user authentication. They were first used in computers at MIT in the mid-60s [30], but their use goes back to ancient times in the Roman military.

2.4.1 Authentication Model

Password based authentication is a simple efficient authentication model, based on a shared secret between a user and a system. The secret (password) is often used in a combination with a user ID. The password itself is a set of characters memorised by the user, and inputted via a keyboard.

Using NIST Digital Identity Guidelines terminology [25], the password and the user ID are issued as a credential and an authenticator to the applicant after successful enrolment by the CSP. A claimant then uses the credentials to authenticate with the verifier, as to establish an authenticated session with the RP.

2.4.2 Security

The threat model needs to account for a variety of attack vectors like network conditions (data confidentiality), integrity of host systems. Some attack vectors are specific to password based authentication, with how passwords are chosen, handled and stored.

Common password based authentication systems over the web rely on the user sending a plain-text password over a secure HTTPS connection, and the server verifying the password.

The simplicity that makes passwords effective is also a big security downside. Because passwords are supposed to be memorised and the proliferation of different websites requiring them, users tend to pick password that are easier to remember and reuse passwords across different websites [16]. Many websites also don't properly handle and store passwords, allowing attackers to "learn" about users passwords in case of a security breach.

2.4.3 Security Attacks

Attacks can be according to NIST [25] classified as *o*nline or *o*ffline, based on wether the attacker is directly interacting with an authentication system or not.

Online Password Attack A form of an *active attack*, where an attacker is attempting bypass authentication by directly interacting with the system. These attacks are usually very noisy, making it easy for an authentication system to detect an attack is happening, and prevent it. This makes online attacks much less effective than offline ones.

Popular methods of online password attacks are *password spraying* and *credential stuffing*, both of which utilise information from data breaches, like username and password combinations, or lists of most commonly used passwords.

Offline Password Attack A form of a passive attack where an attacker is able to analyse data in a system he controls. Data was obtained by the attacker by either theft of file, eavesdropping an authentication protocol or a system penetration.

Password cracking is method of extracting user credentials from data used by the authentication system to verify users credentials.

The success of password cracking is generally determined by two factors, that influence the time required to guess the password.

Password Handling Password handling describes how passwords are stored at rest and used in the verification process.

A naive system might store the passwords or password-equivalent data in plain text and compare them for verification, while simple this system is insecure as user credentials directly are exposed with any unauthorised access.

A common approach today is to use methods of *password hashing* to derive a password digest that is then stored in the database. When verifying the password is hashes again, and the digests are compared.

Using pure hashing functions like SHA family is discouraged because they designed to run fast and can be accelerated with ASIC chips, making them vulnerable to pre-computed hash tables. A better solution are password key-derivation functions. Algorithms utilising hash functions designed with the purpose of being both time-consuming and memory-hard, examples of such tools are Argon2 [7], Scrypt [32] and Balloon [8]. Using an extra value called *salt* [27] prevents attacks with pre-computed hash tables. Because salt is stored alongside password hashes, systems sometimes also utilise a third value called *p*epper, which is the same for all passwords, but stored in a different place from the salt.

Password Strength Measure of information entropy and the difficulty of the password being guesses or brute-forced. Re-using passwords greatly undermines password strength and is what attacks like credential stuffing rely on. Have I Been Pwned [29] catalogs 613,584,248 passwords recovered from data breaches, while CrackStation [28] lists a collection of 1,493,677,782 words used for password cracking.

3 Extensible Authentication Protocol

Extensible Authentication Protocol [1] (EAP) is a general purpose authentication framework, designed for network access authentication, where IP might not be available. It runs directly over the data link layer such as PPP [40] and IEEE 802.

EAP defines a set of messages that support negotiation and execution of a variety of authentication protocols.

3.1 Overview

EAP is a two-party protocol between a *peer* and an *authenticator* at the each end of a link. In the protocol the peer is authenticating with the authenticator.

The protocols is initiated by the authenticator sending a message to the peer, they exchange messages until the authenticator can authenticate the user or not.

3.2 Messages

1	1	2	1	$n \leq 2^{16}$
Code	Identifier	Length	Type	Type-Data

3.2.1 Code Field

The code field determines who the packet is intended for and how or even should the recipient respond.

Code	Name
1	Request
2	Response
3	Success
4	Failure

Request and Response Packets *Request* packets are sent from by the authenticator to the peer. The peer processes the packet and sends back a *response* packet to the authenticator. The response packet needs to have the same identifier as the request packet.

The authenticator will discard response packets that don't have a *matching* identifier with the request packet. The peer does not re-transmit response packets, but relies on the authenticator to re-transmit a request packet after some time if the matching response is lost.

Success and Failure Packets After the authenticator authenticates the peer he sends a *success* packet to the peer. If the peer cannot be authenticated, the authenticator will send a *failure* packet. Both packets signal the end of the authentication process and the peer doesn't need to respond to them.

3.2.2 Request Types

The *t*ype field of a packet indicates the format of the type-data field and the methods used to process the data. First three types are special purpose types.

Type	Name
1	Identity
2	Notification
3	Nak
254	Expanded Type

Identity Used to query the identity of the peer. The type is often used as an initial message from the authenticator the peer.

Notification Used to convey an informative message to the peer, by the authenticator. Usage of this type is entirely optional.

Nak Used only as a response to a request, where the desired authentication type is not available. The peer includes desired authentication methods, indicated by their type number. This type is also referred to as Legacy Nak, when compared to *Expanded Nak* (sub-type of the Expanded Type).

Expanded Type The type field in the EAP packet is 1 octet long, and can represent 256 distinct values. *Expanded types* expand the space for available method types by adding a *Vendor-ID* field and a *Vendor-Type*.

1	1	2	1	3	4	*n*
Code	Identifier	Length	Type	Vendor-ID	Vendor-Type	Data

When a peer does not support the authentication method requested in an Expanded Type request it needs to respond with an Expanded Nak response. If the peer lack support for expanded types, it needs to respond with a legacy nak.

Authentication Methods The remaining types correspond to different authentication methods. According to IANA [35] 49 authentication methods have been assigned type numbers. The original RFC [1] already assigned 3 authentication protocols.

Type	Name
4	MD5-Challenge
5	One-Time Password
6	Generic Token Card

Some notable examples are EAP-TLS [39], EAP-PSK [6]. EAP SRP-SHA1 [13] is especially interesting as the peer uses a ZKP to authenticate itself.

3.3 Pass-Through Behaviour

An authenticator can acts as a *Pass-Through Authenticator*, by using the authentication services of a *backend authentication server*. In this mode of operation the authenticator is relaying the EAP messages between the peer and the backend authentication server. In IEEE 802.1x the authenticator communicates with a RADIUS server [15].

3.4 IEEE 802.1x

IEEE 802.1x is a port based network access control standard for LAN and WLAN. It is part of the IEEE 802.11 group of network protocols.

IEEE 802.1x defines an encapsulation of EAP for use over IEEE 802 as EAPOL or ”EAP over LANs”. EAPOL is used in widely adopted wireless network security standards WPA2. In both WPA2-Personal and WPA2-Enterprise, EAPOL is used for communication between the supplicant and the authenticator.

With WPA2-Enterprise, the authenticator functions in a pass-through mode and uses a RADIUS server to authenticate the supplicant. EAP packets between the authenticator and the authentications server (RADIUS) are encapsulated as RADIUS messages [2, 14, 15]

4 Zero-Knowledge Proofs

4.1 Introduction

Traditional theorem proofs are logical arguments that establish truth through inference rules of a deductive system based on axioms and other proven theorems. *Zero-Knowledge Proofs* (ZKPs) are compared to traditional proofs probabilistic meaning they *"convince"* the verifier with a small margin of error.

They were first defined by Goldwasser, Micali and Rackoff in [24] in a paper published in 1985. They proposed a proof system as a two-party protocol between a *prover* and a *verifier*. It relies on the computational difficulty of the quadratic residuosity problem (QRP).

4.1.1 The Strange Cave of Ali Baba

A famous example of a zero-knowledge proof protocol made by [33] is The Strange Cave of Ali Baba.

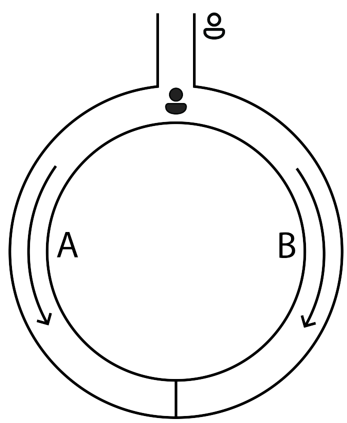

Figure 2: The Strange Cave of Ali Baba

Ali Baba's cave has a single entrance, that splits into two tunnels that meet in the middle where there is a door that can only be opened with a secret passphrase.

Peggy (or Prover) wants to prove to Victor (or Verifier) that she knows the secret passphrase, but she doesn't want to revel the secret nor does she want to reveal her knowledge of the secret to anyone else besides Victor.

To do this they come up with a scheme. Victor turns away from the entrance of the cave, so he cannot see Peggy, as she enters the cave and goes into one of the tunnels at random. Victor then turns around and tells Peggy which tunnel to come out of. Peggy knowing the secret can pass through the door in the middle and emerge from the tunnel requested.

If Peggy didn't know the secret she could still convince Victor, by entering the correct tunnel by luck. But since Victor is choosing the tunnel at random, Peggy's chance of picking the correct tunnel is 50%. If Victor were to repeat the process n time, her chances of fooling him become arbitrarily small (2^{-n}).

With this process Victor can be convinced that Peggy really knows the secret with a very chance ($1-2^{-n}$).

Further more any third party observing the interaction cannot be convinced of the validity of the proof because it cannot be assured that the interaction was truly random. For example, Victor could have told Peggy his questions in advance, so Peggy would produce a convincing looking proof.

4.2 Applications

Most commonly ZKPs were used in authentication and identification systems, as a way to prove knowledge of a secret. Recently however there have been a number of new applications in the cryptocurrency and digital identity spaces.

The cryptocurrency Zcash uses a *non-interactive zero-knowledge protocol* zk-SNARK [9] to prove the validity of transactions, without revealing anything about the recipients nor the amount sent.

The cryptocurrency Monero uses a ZKP protocol Bulletproofs [10], to achieve anonymous transactions.

Idemix [12] an anonymous credential system for interaction between digital identities relies on CL-signatures [11] to prove ownership of a credential offline, without the issuing organisation. Idemix has been implemented in the open-source Hyperledger Indy project.

4.3 Interactive Proof Systems

Interactive proof systems are proof systems between a prover and a verifier, which exchange messages to decide on the validness of the proof. The prover is a computationally unbounded polynomial time Turing machine and the verifier is a probabilistic polynomial time Turing machine.

The properties of *completeness* and *soundness* define an interactive proof system.

Completeness Any honest prover can convince the verifier with overwhelming probability.
For each $k \in \mathbb{N}$ and sufficiently large n;

$$\Pr[x \in L; P(x) = y; V(y) = 1] \geq 1 - \frac{1}{n^k}$$

Soundness Any verifier following the protocol will reject a cheating prover with overwhelming probability.
For each $k \in \mathbb{N}$ and sufficiently large n;

$$\Pr[x \notin L; P(x) = y; V(y) = 0] \geq 1 - \frac{1}{n^k}$$

4.3.1 Interactive Polynomial Time Complexity

Any problem solvable by an interactive proof systems is in the class of **IP**.

4.3.2 Other Variants of Interactive Proof Systems

Arthur-Merlin protocol Problems in the class **AM**, an Arthur-Merlin protocol [4] is an interactive protocol similar to IP, with the difference in that its a *public-coin protocol*. Meaning that verifiers internal state is visible to the prover, while in IP the state is hidden.

Multi Prover Interactive Proofs **MIP** [5] is a more powerful model, utilising two provers that communicate with a single verifier. This models has been build to address the shortcomings of IP. MIP proved that every problem has a ZKP system, without the assumption that one-way functions exist.

4.4 Knowledge Complexity

Zero-knowledge proof systems prove the membership of x in language L, without revealing any additional knowledge (e.g why is $x \in L$).

The essence of zero-knowledge is the idea that what the verifier *sees* is indistinguishable from what can be easily *simulated* on public inputs. The term *knowledge complexity* quantifies the degrees of indistinguishability of different languages and proof constructions.

4.4.1 Indistinguishability

Indistinguishability describes degrees of an ability to distinguish between two random variables U, V.

Let $U = \{U(x)\}$ and $V = \{V(x)\}$ be two families of random variables, where x is from a language L, a subset of $\{0,1\}^*$.
An algorithm $A(x)$ is given a random sample x from either distribution and will output either 1 or 0, depending which distribution it determines the sample originated from. Distributions become "indistinguishable" as the outputs of the algorithm become uncorrelated to the origin of the sample.

By bounding the *size* of the sample and the *time* given to the algorithm we can obtain different notions of indistinguishability.

Equality If $U(x)$ and $V(x)$ are equal, outputs of a computationally unbounded algorithm will remain uncorrelated with the origin of the sample.

Statistical Indistinguishability Two random variables are statistically indistinguishable, when the algorithms outputs remain uncorrelated with the origin, given an arbitrary amount of time and a poly-bounded sample size.

Let $L \subset \{0,1\}^*$ be a language, $U(x)$ and $V(x)$ are statistically indistinguishable on L if,

$$|\Pr[A(x,U) = 1] - \Pr[A(x,V) = 1]| < |x|^{-c}$$

for $\forall c > 0$, and sufficiently long $x \in L$.

Computational Indistinguishability Two random variables are computationally indistinguishable, when the poly-time bounded algorithms outputs remain uncorrelated with the origin, given a poly-bounded sample size.

Let $L \subset \{0,1\}^*$ be a language, poly-bounded families of random variables $U(x)$ and $V(x)$ are computationally indistinguishable on L if for all poly-sized family of circuits C, $\forall c > 0$, and a sufficiently long $x \in L$

$$|\Pr[C(U,x) = 1] - \Pr[C(V,x) = 1]| < |x|^{-c}$$

4.4.2 Approximability of Random Variables

The notion of approximability described the degree to which a random variable $U(x)$ can be "generated" by a probabilistic Turing machine M, generating a probability distribution $M(x)$.

A random variable $U(x)$ is *perfectly approximable* if there exists a probabilistic Turing machine M, such that for $x \in L$, $M(x)$ is *equal* to $U(x)$.
$U(x)$ is statistically or computationally approximable if $M(x)$ is statistically or computationally indistinguishable from $U(x)$.

Generally speaking when saying a family of random variables $U(x)$ is *approximable* we mean that it is *computationally* approximable.

4.4.3 Definition of Zero-Knowledge

Zero-knowledge is a degree of protocols knowledge complexity at which no meaningful information can be extracted by the verifier or any third party observer.

A protocol is zero-knowledge if the verifiers "view" is approximable by a simulator S. A verifiers view is all data that was exchanged with the prover, a cheating verifier's view might have extra information (e.g a history of previous interactions).

A protocols is perfectly zero-knowledge if the view is perfectly approximable for all verifiers. Statistical or computational zero-knowledge is obtained by statistical or computational approximability.

5 Languages with Zero-Knowledge Proof Systems

The zero-knowledge property of interactive proofs is determined by the language the proof exists for. The choice of language also determines the ZKPs practical applicability.

The original ZKP protocols [24] were proposed for the languages of Quadratic Residuosity problem (QRP) and Quadratic Non-Residuosity Problem (QNRP). Other simpler protocols are also based on the Discrete Logarithm problem [41] and Graph Isomorphism Problem [23].

It has been proven in [22] that every language in **NP** has a ZKP system.

ZKP and interactive protocols have also been used as a tool for studying language complexity [37].

In this thesis we are focusing the language QRP.

5.1 Zero-Knowledge Proof of Quadratic Residuosity Problem

Quadratic Residuosity Problem was used in the original ZKP protocol in the founding paper [24]. QRP has a *perfect* zero-knowledge proof system.

QRP is much older than the [24] paper, it was first described by Gauss in 1801 [20].

5.1.1 Quadratic Residues

[3] Quadratic residues come from modular arithmetic, a branch of number theory.

For $a,n \in \mathbb{Z}$, $n > 0$, $\gcd(a,n) = 1$. a is a *quadratic residue* if $\exists x : x^2 \equiv a \pmod{n}$, otherwise a is a *quadratic non-residue*.

When n is an odd prime, a is a quadratic residue modulo n, if and only if.

$$a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$$

Legendre Symbol $\left(\frac{a}{p}\right)$ simplifies computations with quadratic residues.

If p is an odd prime then,

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue modulo } p \\ 0 & \text{if p | a} \\ -1 & \text{otherwise} \end{cases}$$

Jacobi Symbol A generalised definition of the Legendre symbol $\left(\frac{a}{m}\right)$, to allow the case where m is any odd number.

If $m = p_1 p_2 \cdots p_n$, where p_i are odd primes, then

$$\left(\frac{n}{m}\right) = \left(\frac{n}{p_1}\right)\left(\frac{n}{p_2}\right)\cdots\left(\frac{n}{p_n}\right)$$

5.1.2 Prime Factorization

[3] The *Fundamental Theorem of Arithmetic* states that for each integer $n > 1$, exist primes $p_1 \leq p_2 \leq \cdots \leq p_r$, such that $n = p_1 p_2 \cdots p_r$.

The process of prime factorization is a decomposition of a number n to its prime factors $p_1 p_2 \cdots p_r$.

Currently no efficient algorithm exists for prime factorization. The problem is especially hard when factoring *semiprimes*, a product of two prime numbers. This hardness of this problem is used as a core building block in modern asymmetric cryptography like RSA [34].

5.1.3 Quadratic Residuosity Problem

Given a, semiprime $n = pq$, where p and q are unknown different primes, and Jacobi symbol $\left(\frac{a}{n}\right) = 1$.
Determine wether a is a quadratic residue modulo n or not.

The Jacobi Symbol can be efficiently computed using the *Law of Quadratic Reciprocity*, but it does not always tell us if a is quadratic residue modulo n or not.

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$$

If $\left(\frac{a}{n}\right) = 1$ then a is a quadratic residue both modulo p and q, or a is a quadratic non-residue both modulo p and q. To know wether a is a quadratic residue modulo

n or not, we would have to know the prime factorization p, q of n.
If $\left(\frac{a}{n}\right) = -1$ we know a is a quadratic non-residue modulo p or q.

5.1.4 ZKP Protocol for the Quadratic Residuosity Problem

n	Semiprime, where $\left(\frac{x}{n}\right) = 1$
x	Public input, where $x = w^2 \pmod n$
w	Provers private input

Prover		Verifier
$u \leftarrow \mathbb{Z}_n^*; y = u^2 \pmod n$	$\xrightarrow{y}$	
	$\xleftarrow{b}$	$b \leftarrow_R \{0,1\}$
$z = uw^b \pmod n$	$\xrightarrow{z}$	verify $z^2 = yx^b \pmod n$

This protocol is repeated m times, for a probability of error of $\frac{1}{2^m}$.

5.2 Computational Complexity Classes

5.2.1 Non-deterministic Polynomial Time

NP is a class of problems solvable by a non-deterministic Turing machine in polynomial time. Or rather proof of any language in NP can be verified by a deterministic Turing machine in polynomial time.

Article [22] proved that every language in NP has a zero-knowledge proof system, by creating a ZKP protocol for the Graph 3-Colouring problem (3-COL).

Minimum Colouring Problem is a problem in graph theory, of what is the minimal *k proper* colouring of a graph, where no adjacent vertices are the same colour. An instance of ($k = 3$) colouring (3-COL) is proven to be *NP-Hard* because a polynomial reduction exists from *Boolean-Satisfiability problem* (3-SAT) to 3-COL [31]. According to Cook's theorem [17] 3-SAT is *NP-Complete*, and any language in $L \in NP$ can be reduced to and instance of 3-SAT. Furthermore because polynomial reductions are *transient*, any language $L \in NP$ can be reduced to an instance of 3-COL.

5.2.2 Bounded-Error Probabilistic Polynomial Time Languages

BBP is a class of problems that can be verified by a probabilistic Turing machine in polynomial time.

Trivially every language in BPP has a ZKP system, where the prover sends nothing to the verifier, the verifier checks the proof of $x \in L$ and outputs a the verdict.

5.3 Alternative Composition of Zero-Knowledge Proofs

Zero-Knowledge Proofs can alternatively be composed in parallel as compared to sequential composition in [24]. Parallel composition is very interesting practically as it can help reduce the inefficiencies of communication between the prover and the verifier, especially over high latency networks.

In [21] they proved that only languages in BPP have 3-round interactive proofs that are zero-knowledge.

The QRP is not believed to be in BPP, so a parallel composition of QRP has weaker notion of zero-knowledge.

6 Password-based Authentication using ZKP of Quadratic Residuosity Problem

One of the original ZKP protocol proposed in [24] was based on the quadratic residuosity problem.

The protocol can be used as a password-based authentication protocol, where the proof proves the possession of a password. The protocol can further enhanced by similar protocols, to make it meet the security standards of modern password-based authentication systems.

6.1 Original Protocol

n	Semiprime, where $\left(\frac{x}{n}\right) = 1$
x	Public input, where $x = w^2 \pmod{n}$
w	Password

	Prover		Verifier
1	$u \leftarrow \mathbb{Z}_n^*; y = u^2 \pmod{n}$	$\xrightarrow{y}$	
2		$\xleftarrow{b}$	$b \leftarrow_R \{0,1\}$
3	$z = uw^b \pmod{n}$	$\xrightarrow{z}$	verify $z^2 = yx^b \pmod{n}$

This protocol is repeated m times, for a probability of error of $\frac{1}{2^m}$.

6.2 Security

The protocol is secure against active attacks like masquerading and replay-attacks. Zero-knowledge also makes it secure against eavesdropping.

The main issue with the protocols as a password based authentication method is vulnerability to dictionary attacks and attacks pre-computed tables.

6.2.1 Password Cracking Vulnerability

The input x is used by the verifier to verify the witness, it is computed from the private input w as $x = w^2 \pmod{n}$. The provers private input w is the password.

The need of the verifier to access the raw value of x prevents the authentication system from processing x with modern password key-derivation methods. This creates a vulnerability for attacks with pre-computed tables. An attacker can pre-compute the values of x and compare them with the stored x data by the verifier.

6.2.2 Prover Password Key-Derivation

To utilise PKDF, we need to apply it to derive the provers private input w. Instead of the password being used directly as w, the password is processed by a PKDF, and the derivation is used as w.

This approach is similar to the one used in [41] the Secure Remote Password protocol. Using a KDF H, a random salt s and password P, we can derive w and x.

$$w = H(P, s)$$
$$x = w^2 \pmod{n}$$

6.3 Protocol

Using the terminology in NIST Digital Identity Guidelines [25]. To draw parallels between this terminology and the terminology used in the ZKP-QRP [24]. The Prover is the Claimant and Applicant, and the Verifier is the Authenticator ant the CSP.

Values

q, p	Primes, where $q \neq p$
n	Semiprime modulus, where $n = qp$
P	Credential password
I	Credential identifier
H	PKDF
s	Salt
w	Password hash, where $w = H(P, s)$
x	Integer, where $x = w^2 \pmod{n}$

Enrolment In the enrolment process the CSP provides the n modulo value to the Applicant. The Applicant generates a random salt s and computes a private w

value from the password $P; w = H(P,s)$. Applicant next computes $x = w^2 \pmod n$ and submits the identifier I, x, s to the CSP.

	Applicant		CSP
1		$\xleftarrow{n}$	
2	$s \leftarrow_R \mathbb{Z}$ $w = H(P,s)$ $x = w^2 \pmod n$	$\xrightarrow{I,x,s}$	

CSP binds x and s as the authenticator to the credential I.

Authentication Authentication happens in two part, in the first part required data is exchanged between the Claimant and the Authenticator. The Claimant identifies himself and the Authenticator provides the semiprime modulus n and the salt s. The second part of the protocol is the ZKP-QRP [24] protocol executed between the Claimant and the Authenticator.

First Part (Setup) The Claimant sends an identifier I to the Authenticator, which responds with modulo n and the salt s. The Claimant uses both values to compute the private input w of the ZKP-QRP protocol.

	Claimant		Authenticator
1		$\xrightarrow{I}$	
2	$w = H(P,s)$	$\xleftarrow{n,s}$	

Second Part (Verification) This part is same as the ZKP-QRP protocol described in the [24].

	Claimant		Authenticator
1	$u \leftarrow_R \mathbb{Z}_n^*$ $y = u^2$	$\xrightarrow{y}$	
2		$\xleftarrow{b}$	$b \leftarrow_R \{0,1\}$
3	$z = uw^b \pmod n$	$\xrightarrow{z}$	verify $z^2 \equiv yx^b \pmod n$

The second part is repeated m times, for a probability of error of $\frac{1}{2^m}$

7 PBA Using ZKP-QRP Implemented as an EAP Method

To define an EAP method for the PBA-ZKP-QRP protocol, we need to define the protocol execution between a peer and the verifier, by defining message subtypes, their data format, and rules for handling them. We also explore different approaches of mapping between PBA-ZKP-QRP message pairs and EAP messages, and their performance.

7.1 EAP Packet Format

An EAP packet is n octets long.

1	1	2	1	1	$n-6$
Code	Identifier	Length	Type	Subtype	Subtype Data

Code The code field is one octet

1 Request
2 Response

Identifier The identifier field is one octet, and is being used to match request and response packets.

Length Two octets Subtypelong, used to indicate the length of the EAP packet.

Type One octet long.

84 EAP PB-ZKP-QRP

Subtype One octet long

1 SETUP
2 ZKP-QRP

The subtype format describes only the contents of the *subtype data* field.

7.1.1 Subtype 1 Request

EAP Subtype 1 request must be sent after obtaining the peers identity. The identity can be acquired with the EAP-Identity (Type 1) packet, or determined somehow otherwise.

The peers identity is used to look up the password salt s and semiprime modulus n.

1	$4 \leq n \leq 255$	$64 \leq m$
Salt Length	Salt	Semiprime Modulus

Salt Length A single octet for the length of the *s*alt field in octets.

Salt A random salt value, should be from 4 octets to 255 octets long. The max length is determined by the max number able to be encoded in the *salt length* field.

Semiprime Modulus Fills the rest of the message to the length specified by the *l*ength field in the EAP header. Should be at least 64 octets (512 bits).

7.1.2 Subtype 1 Response

The request of this subtype serves to complete the setup phase of the protocol, at the same time the response already includes the y value required at the start of each cycle of the second part of the protocol.

n
Square y

Square y Computed by the peer, as $y = u^2$, where $u \leftarrow_R \mathbb{Z}_n^*$. Fills the remainder of the message in n octets.

7.1.3 Subtype 2 Request

1
Random Bit b

Random Bit b A single-bit, at the right-most place. The bit value is randomly chosen by the authenticator. 1 octet long.

7.1.4 Subtype 2 Response

1	$n \leq 255$	m
Witness Length	Witness z	Square y

Witness Length A field one octet in length. Determines the length of the Witness field in octets.

Witness Fields length is limited by the max value of the *witness length* field at 255 octets. The witness z is computed by the peer, as $z = uw^b$, where u was generated for the subtype 1 response, bit b was provided in the request, and w is the provers private input.

Square y Field fills up the remainder of the message. Square y is the same value as in the subtype 1 response. It is generated and sent in the n-th cycle, to help verify the witness in the $(n+1)$-th cycle. Same rules apply as when generating the y value if the response to subtype 1 request.

Verification When authenticator receives the *subtype 2 response*, it checks the witness, by verifying $z^2 \equiv yx^b \pmod{n}$. If verification fails the authenticator should send a *failure* message to the peer, and he authentication should be terminated. After successfully verifying the witness, the authenticator can decided to continue the protocol by sending a *subtype 2 request* or decide to authenticate the user, by sending a *s*uccess message. The authenticator can decide an authentication is successful when the protocol reaches a desired confidence of $1 - \frac{1}{2^m}$, by iterating for m times.

7.2 Optimisations

EAP is a lock-step protocol, the authenticator and the peer exchange request and response messages.

A naive mapping of PBA-ZKP-QRP messages to EAP packets yields 3 new request/response pairs. We can reduce the amount of new pairs to 2 instead of 3, by interlacing data shared in each pair. This way we obtain a faster performance by reducing the number of packet needed to be exchanged.

7.2.1 Naive Map

Pair	Peer	$\leftrightarrow$	Authenticator
1		$\xleftarrow{s,n}$	
		$\xrightarrow{\textvisiblespace}$	
2		$\xleftarrow{\textvisiblespace}$	
		$\xrightarrow{y}$	
3		$\xleftarrow{b}$	
		$\xrightarrow{z}$	

Pair 1 Exchanged once after the authenticator obtaining the peers identity. The authenticator sends the salt s and semiprime modulus n to the peer, in order for the peer to compute the private input w. Peers response serves as an acknowledgement of a successful setup.

This pair corresponds to the *setup* part of the protocol.

Pair 2 The authenticator requests the peer to generate the *square* value y and share it in the response.

This pair corresponds to the ZKP-QRP part of the protocol and is repeated for m times.

Pair 3 The authenticator requests the peer to compute the *witness* value z, with the value of the random bit b in the request data.

This pair corresponds to the ZKP-QRP part of the protocol and is repeated for m times.

Performance With this mapping a successful protocol run of m iterations with a probability of error of 2^{-m}, would require a minimum of $4m+3$ packet exchanges.

Packets exchanged	Type
2	Pair 1
$2m$	Pair 2
$2m$	Pair 3
1	Type 2 (Success)

7.2.2 Interlaced Data Mapping

Pair	Peer $\leftrightarrow$ Authenticator
1	$\xleftarrow{\text{s, n}}$
	$\xrightarrow{y_1}$
2	$\xleftarrow{b}$
	$\xrightarrow{z, y_{n+1}}$

Pair 1 Exchanged once after the authenticator obtains the peers identity. The authenticator sends the salt s and semiprime modulus n to the peer, in order for the peer to compute the private input w. Peer computes the square value y and sends it in the response.

The main difference with the naive mapping is that the peer responds prematurely with y, instead of in the response to naive pair 2. This is possible and valid, because the semiprime modulus value n required to compute y, is provided in the pair 1 request.

Pair 2 The authenticator already has the square value y, and sends a request with a random bit b. The peer computes sends the *witness* z and the square value y_{n+1}, used in the next iteration of the protocol

This is possible because the computation of square value y is only dependent on the modulus n, which is provided in the request pair 1.

Performance With this mapping a successful protocol run of m rounds with an error rate 2^{-m}, would require a minimum of $2m+3$ packet exchanges.

Packets exchanged	Type
2	Pair 1
$2m$	Pair 2
1	Type 2 (Success)

Comparing the performance of both mappings, the interlaced mapping requires half as many exchanges for the same m rounds of protocol.

$$\lim_{1\to\infty} \frac{2m+3}{4m+3} = \frac{1}{2}$$

7.3 Security

EAP PB-ZKP-QRP is resistant to passive attacks to over-the-wire information, eavesdropping, active attacks and offline attacks with pre-computed tables/rainbow tables.

The protocol does not enable mutual authentication, nor helps in deriving a session key that can be used for data encryption.

References

[1] Bernard Aboba, Larry Blunk, John Vollbrecht, James Carlson, Henrik Levkowetz, et al. Extensible authentication protocol (eap). 2004.

[2] Bernard Aboba and Pat Calhoun. Radius (remote authentication dial in user service) support for extensible authentication protocol (eap). Technical report, RFC 3579, September, 2003.

[3] George E Andrews. *Number theory*. Courier Corporation, 1994.

[4] László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429, 1985.

[5] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 373–410. 2019.

[6] Florent Bersani and Hannes Tschofenig. The eap-psk protocol: A pre-shared key extensible authentication protocol (eap) method. Technical report, RFC 4764, January, 2007.

[7] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: new generation of memory-hard functions for password hashing and other applications. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 292–302. IEEE, 2016.

[8] Dan Boneh, Henry Corrigan-Gibbs, and Stuart Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 220–248. Springer, 2016.

[9] Sean Bowe, Ariel Gabizon, and Matthew D Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. In *International Conference on Financial Cryptography and Data Security*, pages 64–77. Springer, 2018.

[10] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[11] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques*, pages 93–118. Springer, 2001.

[12] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, 2002.

[13] J Carlson, B Aboba, and H Haverinen. Eap srp-sha1 authentication protocol (draft-ietf-pppext-eap-srp-03. txt). *Network Working Group, Internet Draft*, 135:136–137.

[14] Jyh-Cheng Chen and Yu-Ping Wang. Extensible authentication protocol (eap) and ieee 802.1 x: tutorial and empirical experience. *IEEE communications magazine*, 43(12):supl–26, 2005.

[15] Paul Congdon, Bernard Aboba, Andrew Smith, Glen Zorn, and John Roese. Ieee 802.1 x remote authentication dial in user service (radius) usage guidelines. *RFC*, 3580:1–30, 2003.

[16] Art Conklin, Glenn Dietrich, and Diane Walz. Password-based authentication: a system perspective. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 10–pp. IEEE, 2004.

[17] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[18] Federal Financial Institutions Examination Council. Authentication in an internet banking environment. *FFIEC gencies (August 2001 Guidance)*, 2005.

[19] ECB ECB. Recommendations for the security of internet payments. Technical report, Tech. Rep. January, 2013.

[20] Carl Friedrich Gauss. *Disquisitiones arithmeticae*. 1801.

[21] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[22] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. volume 263, pages 171–185, 08 1986.

[23] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 285–306. 2019.

[24] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC 85, page 291304, New York, NY, USA, 1985. Association for Computing Machinery.

[25] Paul A Grassi, Michael E Garcia, and James L Fenton. Nist special publication 800-63-3 digital identity guidelines. *National Institute of Standards and Technology, Los Altos, CA*, 2017.

[26] Dick Hardt et al. The oauth 2.0 authorization framework. Technical report, RFC 6749, October, 2012.

[27] Taylor Hornby. Salted password hashing-doing it right. *Code Project: For those who code*, 2016. .

[28] Taylor Hornby. Crackstation's password cracking dictionary. 2019. .

[29] Troy Hunt. Have i been pwned. *Last retrieved*, 2021.

[30] Robert McMillan. The world's first computer password? it was useless too. 2012. https://www.wired.com/2012/01/computer-password/.

[31] Lalla Mouatadid. Introduction to complexity theory: 3-colouring is np-complete.

[32] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. *IETF Draft URL: http://tools. ietf. org/html/josefsson-scrypt-kdf-00. txt (accessed: 30.11. 2012)*, 2016.

[33] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michal Quisquater, Louis Guillou, Marie Guillou, Gad Guillou, Anna Guillou, Gwenol Guillou, Soazig Guillou, and Thomas Berson. How to explain zero-knowledge protocols to your children. pages 628–631, 08 1989.

[34] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[35] Joseph Salowey. Extensible authentication protocol (eap) registry, method types. *IANA Extensible Authentication Protocol (EAP) Registry*, 2004.

[36] R Schaeffer. National information assurance (ia) glossary. *CNSS Secretariat, NSA, Ft. Meade*, 2010.

[37] Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

[38] Robert Shirey. Internet security glossary, version 2. Technical report, RFC 4949, August, 2007.

[39] Dan Simon, Bernard Aboba, Ryan Hurst, et al. The eap-tls authentication protocol. *RFC 5216*, 2008.

[40] William Simpson. *RFC1661: the point-to-point protocol (PPP)*. RFC Editor, 1994.

[41] Thomas D Wu et al. The secure remote password protocol. In *NDSS*, volume 98, pages 97–111. Citeseer, 1998.