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

Jakob Povsic

2020

Contents

1 Abstract

In this thesis I will introduce the notion of zero-knowledge proofs, their variants and formal definitions. In the second part of the thesis I will present a protocol for password based authentication based on the zero-knowledge proof of quadratic residuosity. In the last part of the thesis I will document the implementation of the protocol as an EAP authentication method.

2 Introduction

Something about how privacy and security are becoming ever more important as our world is becoming ever more digitalised and connected.

2.1 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 [BGG18] to prove the validity of transactions, without revealing anything about the recipients nor the amount sent.

The cryptocurrency Monero uses a ZKP protocol Bulletproofs [BBB$^+$18], to achieve anonymous transactions.

Idemix [CVH02] an anonymous credential system for interaction between digital identities relies on CL-signatures [CL01] to prove ownership of a credential offline, without the issuing organisation. Idemix has been implemented in the open-source Hyperledger Indy project.

2.2 Example

A famous example of a zero-knowledge proof protocol made by [QQQ$^+$89] is 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.

3 Zero-Knowledge Proofs

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 [GMR85] 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).

3.1 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;

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

3.1.1 Interactive Polynomial Time Complexity

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

3.1.2 Other Variants of Interactive Proof Systems

Arthur-Merlin protocol Problems in the class **AM**, an Arthur-Merlin protocol [Bab85] 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** [BOGKW19] 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.

3.2 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.

3.2.1 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 polynomial time algorithms outputs remain uncorrelated with the origin, given a poly-bounded sample size.

Computational Indistinguishability Two random variables are computationally indistinguishable, if judges verdict remains meaningless given a polynomial sized sample and polynomial amount of time.

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 $x \in L$

$$|P(U,C,x) - P(V,C,x)| < |x|^{-c}$$

Any two families that are *computationally indistinguishable* are considered *indistinguishable* in general.

3.2.2 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.

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.

3.2.3 Definition of Zero-Knowledge

The zero-knowledge property is addressing the absence of meaningful information that can be extracted from the protocol by an honest or an cheating verifier.

The verifiers *view* is all data he sees in the interaction with the prover as well data already possessed by the verifier, for example previous interactions with the prover.

An interactive protocol is *perfectly* zero-knowledge if the verifiers view is *perfectly approximable* for all verifiers. Generally we say an interactive protocol is zero-knowledge when its *computationally* zero-knowledge.

4 Languages with Zero-Knowledge Proof Systems

One of the main components that make Zero-Knowledge proofs work is the encoding of the proof in the *solution* of another ”problem”. The choice of the ”problem” heavily relies on the specific application of the ZKP protocol.

A theoretical term from the computational complexity theory for a ”problem” is *language*. And the ”problem” is the task of proving the membership of x in language L

Alongside specific languages with ZKPs, they have been also studies related to classes of languages defined by their computational complexity.

In this thesis we are focusing on the zero-knowledge proof of quadratic residuosity, but generally ZKP protocols exists for any language in NP [GMW86], assuming one way-functions exist in IP[*]

4.1 Quadratic residuosity problem

The original ZKP protocol described in [GMR85] was based on the *quadratic residuosity problem*. The paper provided a perfect ZKP protocol for QRP and a statistical ZKP protocol for *quadratic non-residuosity problem*.

Quadratic residuosity problem (QRP) is much older than the [GMR85] paper. It was first described by Gauss in 1801 [Gau01]. Quadratic residues come from modular arithmetic a branch of number theory.

Quadratic Residues For $a, n \in \mathbb{Z}$, $n > 0$, a and n are co-prime. a is a *quadratic residue* if $\exists x : x^2 \equiv a \pmod{n}$, otherwise a is a *quadratic non-residue*

Problem Given numbers a and $n = pq$, where p and q are unknown different primes, and $(\frac{a}{n}) = 1$[†], determine wether a is a quadratic residue modulo n or not.

The problem of quadratic residuosity is considered difficult, because prime factorisation is hard.

[*]Class of problems solved by an *interactive proof system*

[†]Jacobi symbol

Protocol

Public inputs $n, x : (\frac{x}{n}) = 1$ and
Provers private input $w : x \equiv w^2 \pmod{n}$

- P $\rightarrow$ V: Prover chooses random $u \leftarrow \mathbb{Z}_n^*$ and sends $y = u^2$ to the verifier.
- P $\leftarrow$ V: Verifier chooses $b \leftarrow_R \{0, 1\}$
- P $\rightarrow$ V: If $b = 0$ Prover sends u to the Verifier, if $b = 1$ Prover sends $z = w \cdot u \pmod{n}$.
- Verifier accepts if, $[b = 0], z^2 \equiv y \pmod{n}$ or $[b = 1], z^2 \equiv xy \pmod{n}$ or rejects and halts otherwise.

This protocol is repeated m times.

4.2 Computational Complexity Classes

4.2.1 Bounded-Error Probabilistic Polynomial Time Languages

Or **BBP** in short, is in computational complexity theory a class of problems solvable by a probabilistic Turing machine in polynomial time with a bounded error to at most $1/3$ or $2^{-ck}; c > 0$ for k iterations.

4.2.2 Non-deterministic Polynomial Time

Or **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 polynomial time Turing machine.

In [GMW86] proved that every problem in NP has a zero-knowledge proof system, by describing a ZKP protocol for the Graph 3-Colouring problem (3-COL)

Minimum colouring problem is problem in graph theory, of what is the minimal *k proper* colouring of a graph, so that no adjacent vertices are the same colour.

An instance of 3-COL is proven to be *NP-Hard* because a polynomial reduction exists from *Boolean-Satisfiability problem* (3-SAT) to 3-COL [Mou].

According to Cook's theorem [Coo71] 3-SAT is NP-Complete, and any language in $L \in NP$ can be reduced by a polynomial deterministic Turing machine to 3-SAT. Furthermore because polynomial reductions are *transient*, any language $L \in NP$ can be reduced to an instance of 3-COL.

4.3 Alternative Constructions of Zero-Knowledge Proofs

5 Authentication

As defined by the RFC-4949 [Shi07], authentication is ”The process of verifying a claim that a system entity or system resource has a certain attribute value.” This is a generic definition, and it most frequently applies to the verification of user identity (e.g at login), however assertions can be made and verified about any subject or object.

Two components defined an authentication process.

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.

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.

5.1 The NITS Model for Digital Identity

Digital Identity Guidelines [GGF17] 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*.

Enrolment (and Identity Proofing) The enrolment describes 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 verifier by sharing the credential and the authenticators. The verifier validates binding between the credential and authenticators with the CSP. An authenticated connection is established between the subscriber and the RP after and assertion is provided by the CSP or the verifier to the RP.

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

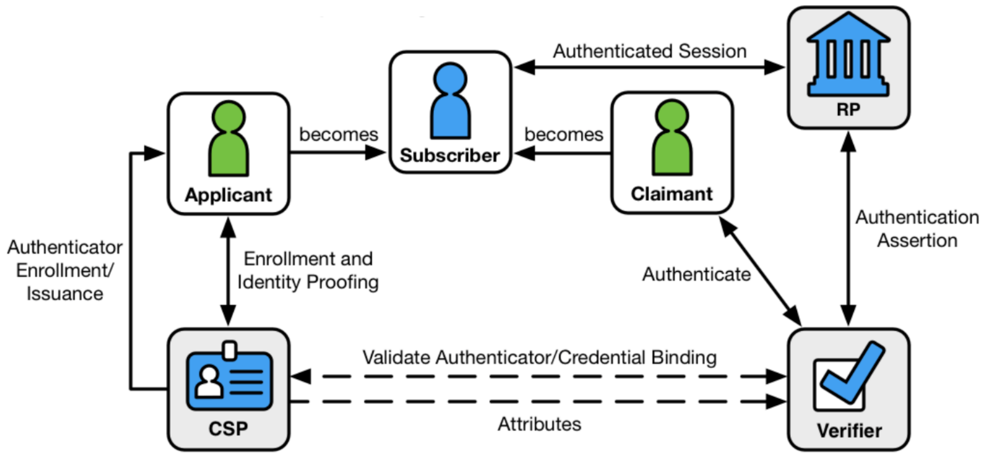

Figure 1: NITS Digital Identity Model

Note on delegation of roles In the digital identity model 1 roles of CSP, verifier and RP 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 [H$^+$12] authentication layer, the resource owner has the roles of applicant, claimant and subscriber. The authorisation server has the roles of CSP and verifier. The OAuth2 client has the role of the RP.

5.2 Authentication Factors

As described in [Cou05] 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 government and financial institutions is, an authentication procedure based two or more authentication factors. Authentication using two or more factors is also referred to as *multi-factor authentication*.

5.3 Password Based Authentication

Passwords are one of the oldest forms of authentication, dating back to ancient times when it was used in the Roman military. It was first used in computers at MIT in the mid-60s [McM12].

Password based authentication is a simple authentication model, based on a shared secret between a user and a system. The secret (password) is used in a combination with a user ID. Most common passwords are a set of characters and are memorised by the user.

Using NIST Digital Identity Guidelines terminology [GGF17], 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.

5.3.1 Security

The simplicity of password-based authentication is also its downside, and the IT community has actively been working to move away from password based authentication and its downsides. While there are many general attack vectors in any authentication system such as network conditions (man-in-the-middle) and the integrity of the system (viruses, key-loggers). The biggest risk specific to a password-based authentication system is choosing, handling and storing the passwords.

Passwords need to be memorised by the user, which creates an incentive for users to pick passwords that are shorter in length and contain less variance [CDW04].

Ways in which adversaries can attack a PBS can be generally categories based on attackers access to ”authenticator data”, NIST [GGF17] classifies *online* and *offline* attacks as.

Online attack An attack where the attacker assumes the role of the claimant with a genuine verifier. A common type of attack is a *guessing attack*, where and adversary repeatedly attempts to authenticate with the verifier by guessing possible passwords, this makes the attack very noisy and easy to detect and mitigate.

Offline attack An 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.

Protecting against an offline attack means making it very expensive for an attacker to guess the password or ”crack the password”. What influences the cost of guessing a password is the expected password entropy and the time to guess a single password.

Password storage An important defence against offline attack are the methods with which passwords are stored and verified. Trivially passwords stored in plaintext or hashed passwords are easy for an attacker to exploit efficiently with methods like pre-computed hash and rainbow tables.

Modern password based security relies on methods of key-derivation that make password cracking both time-consuming and memory-hard [PJ16, BDK16, BCGS16] and using *salt* [Hor16] to make pre-computation attacks infeasible. Key-derivation functions transform the plain text passwords into password hashes for storage. When the system wants to verify a password it repeats the process of key-derivation and compares the output hash with the one in storage.

Password strength *The strength* of a password directly relates to its entropy. The overall cost password cracking can be reduced by orders of magnitude by a week password, because the attacker has a smaller pool of passwords to try. Have I Been Pwned [Hun21] catalogs 613,584,248 passwords recovered from data breaches, while CrackStation [Hor19] lists a collection of 1,493,677,782 words used for password cracking. Any password cracking software will exhaust the list in relatively short time compared to a brute-forcing technique.

6 Extensible Authentication Protocol

Extensible Authentication Protocol [ABV$^+$04] (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 [Sim94] and IEEE 802.

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

6.1 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 protocol is initiated by the authenticator, by sending a *Request* message to the peer, and the peer responds with a *Response* message in a lock-step fashion. The success of the authentication is signalled with a terminal *Success* or *Failure* message.

6.2 Messages

Request and Response Request messages are send from the authenticator to the peer. Request packets have a Type field that indicates what is requested. The peer processes the packet according to the Type field and send a Response of the same Type. The Type of the Request determines the data in the packet.

Success and Failure After a successful completion of an authentication method an authenticator sends a Success packet to peer. A Failure packet is sent if the peer cannot be authenticated with the authenticator.

6.3 Request Types

The Type field of a Request packet indicates what information is being requested. First three types are special purpose types.

6.3.1 Identity

Type 1. Used to query the identity of the peer.

6.3.2 Notification

Type 2. Used to convey a message from the authenticator to the peer.

6.3.3 Nak

Type 3. 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).

6.3.4 Expanded Type

Type 254. 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 (3 octets) and a *Vendor-Type* (4 octets).

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.

6.3.5 Authentication Methods

The remaining types correspond to different authentication methods. According to IANA 49 authentication methods have been assigned Type numbers [Sal04].

The original RFC [ABV$^+$04] already assigned 3 authentication protocols.

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

Some notable examples are EAP-TLS [SAH$^+$08], EAP-PSK [BT07], EAP SRP-SHA1 [CAH]. The IANA list is not accounting for any authentication methods supported with the Expanded Type.

6.4 Pass-Through Behaviour

An authenticator can acts as a ”Pass-Through Authenticator”, relying on authentication services of a *backend authentication server*. In this mode of operation the authenticator is relaying the EAP packets between the peer and the backend authentication server.

In IEEE 802.1x the authenticator communicates with a RADIUS server [CAS+03].

6.5 IEEE 802.1x

IEEE 802.1x is standard for port based network access control 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 (wireless station) and the authenticator (access point).

With WPA2-Enterprise, the authenticator (Access Point) functions in a pass-through mode as uses a RADIUS server for authentication services. EAP packets between the authenticator and the authentications server (RADIUS) are encapsulated as RADIUS messages [AC03, CW05, CAS+03]

7 Password-based Authentication using Zero-Knowledge Proof of Quadratic Residues

The original paper on zero-knowledge proofs outlined a zero-knowledge proof protocol based on the problem of quadratic residues.

7.1 Protocol

Public inputs $n, x : (\frac{x}{n}) = 1$
Provers private input $w : x \equiv w^2 \pmod{n}$

Condition	Prover		Verifier
	$u \leftarrow_R \mathbb{Z}_n^*;\quad y = u^2$	$\xrightarrow{\text{y}}$	
		$\xleftarrow{\text{b}}$	$b \leftarrow_R \{0,1\}$
$b = 0$	$z = u$	$\xrightarrow{\text{z}}$	(verify $z^2 \equiv y \pmod{n}$)
$b = 1$	$z = wu \pmod{n}$	$\xrightarrow{\text{z}}$	(verify $z^2 \equiv xy \pmod{n}$)

This protocol is repeated m times, for a confidence of $1 - 2^{-m}$.

7.2 Security

7.2.1 Offline Attacks

The input x is used by the Verifier to verify the proof, it is computed from the private input w as $x = w^2 \pmod{n}$. The w is the Provers private input and represents the users password in a password-based authentication system.

The problem in an application of this protocols in the storage of w. In modern password-based authentication system, the password is processed by a resource expensive password key derivation function, and the authentication is verified by comparing the derived values. Our protocol prevents us from using key derivation in this way, because without the original or ”plaintext” value x we cannot verify the proofs provided by the Prover.

However storing the ”plaintext” values of x, enables an attacker to easily precompute a table of possible x values.

7.2.2 ?? Client-Side Key Derivation

Our wish is to use a hashing function with salt on the password and use the original value of x for verification. To achieve both requirements, the password is hashed before used to calculate the x used for verification in the authentication process. This approach is similar to the one used in [W$^+$98] the Secure Remote Password protocol. Using a hashing function H, a random salt s and users password P, we can derive w and x.

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

7.3 Protocol

Using the terminology in NIST Digital Identity Guidelines [GGF17]. To draw parallels between this terminology and the terminology used in the ZKP-QRP [GMR85]. The Prover is the Claimant, and the Verifier is the Authenticator.

Values

q, p	Primes, where $q \neq p$
n	Modulus, where $n = qp$
P	Provers password
I	Provers identifier
H	Hashing function
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 and x to the CSP.

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

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

Authentication Authentication happens in two part, in the first part minimal data is exchanged between the Claimant and the Authenticator. The Claimant identifies himself and the Authenticator provides the n and the salt s. In the second part the protocol for ZKP-QRP is 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.

C $\rightarrow$ Au:	Send I
Au $\rightarrow$ C:	Send n, s
C:	Computes $w = H(P,s)$

Second Part (Verification) This part is same as the ZKP-QRP protocol described in the [GMR85].

C $\rightarrow$ Au:	Choose $u \leftarrow_R \mathbb{Z}_n^*$; Send $y = u^2$
Au $\rightarrow$ C:	Choose $b \leftarrow_R \{0,1\}$; Send b
P $\rightarrow$ V $[b = 0]$:	Send u
P $\rightarrow$ V $[b = 1]$:	Send $z = w \cdot u \pmod{n}$
V $[b = 0]$:	Check $z^2 \equiv y \pmod{n}$
V $[b = 1]$:	Check $z^2 \equiv xy \pmod{n}$

The second part is repeated m times, for confidence of $1 - 2^{-m}$.

7.3.1 Security

8 PBA Using ZKP-QRP Implemented as an EAP Method

8.1 EAP Packet Format

An EAP packet is n octets long.

1	1	2	1	1	$n-6$
Code	Identifier	Length	Type	Sub-Type	Sub-Type 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 long, used to indicate the length of the EAP packet.

Type One octet long.

69	EAP PB-ZKP-QRP

Subtype One octet long

1	SETUP
2	ZKP-QR

8.1.1 Subtype 1 Request

EAP Sub-Type 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 modulus n.

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

Salt Length A single octet for the length of the Salt 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.

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

This is the n value in the ZKR-QR protocol, a product of two primes $n = qp$.

8.1.2 Subtype 1 Response

The request of this subtype serves to complete the setup phase of the protocol, while the response already provides 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.

8.1.3 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.

8.1.4 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, the computation depends on the value of the random bit b in the request. If $b = 0$, then $z = u$, where u was generated for the subtype 1 response. If $b = 1$, then $z = w \cdot u$, where u was generated for the subtype 1 response, 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.

8.2 Optimisations

EAP is a lock-step protocol of request response pairs, each packet is first sent by the authenticator as a request, and the peer returns the message as a response.

A naive mapping of 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 can obtain faster performance by reducing the number of packet needed to be exchanged.

Naive Map

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

Pair 1 Exchanged once after the authenticator obtaining the peers identity. The authenticator communicates the salt s and 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, according to the procedure determined by 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)

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 obtaining the peers identity. The authenticator communicates the salt s and modulus n to the peer, in order for the peer to compute the private input w. Peers 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. Trivially we see, that it is possible and valid, because the modulus value n required to compute y, is provided in the pair 1 request.

Pair 2 The authenticator already receiving the square value y can send a request with a random bit b. The peer responds by computing the *witness* z. The peer also computes the square value y_{n+1}, which is 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 established 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.

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

8.3 Security

EAP PB-ZKP-QRP is resistant to passive attacks, as a ZKP protocol by its definition reveals no information, except the validity of the proof to the verifier. It is also resistant to replay attacks.

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

References

[ABV+04] Bernard Aboba, Larry Blunk, John Vollbrecht, James Carlson, Henrik Levkowetz, et al. Extensible authentication protocol (eap). 2004.

[AC03] Bernard Aboba and Pat Calhoun. Radius (remote authentication dial in user service) support for extensible authentication protocol (eap). Technical report, RFC 3579, September, 2003.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429, 1985.

[BBB+18] 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.

[BCGS16] 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.

[BDK16] 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.

[BGG18] 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.

[BOGKW19] 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.

[BT07] Florent Bersani and Hannes Tschofenig. The eap-psk protocol: A pre-shared key extensible authentication protocol (eap) method. Technical report, RFC 4764, January, 2007.

[CAH] 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.

[CAS$^+$03] 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.

[CDW04] 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.

[CL01] 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.

[Coo71] 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.

[Cou05] Federal Financial Institutions Examination Council. Authentication in an internet banking environment. *FFIEC gencies (August 2001 Guidance)*, 2005.

[CVH02] 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.

[CW05] 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.

[Gau01] Carl Friedrich Gauss. *Disquisitiones arithmeticae*. 1801.

[GGF17] 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.

[GMR85] 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.

[GMW86] 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.

[H$^+$12] Dick Hardt et al. The oauth 2.0 authorization framework. Technical report, RFC 6749, October, 2012.

[Hor16] Taylor Hornby. Salted password hashing-doing it right. *Code Project: For those who code*, 2016. .

[Hor19] Taylor Hornby. Crackstation's password cracking dictionary. 2019. .

[Hun21] Troy Hunt. Have i been pwned. *Last retrieved*, 2021.

[McM12] Robert McMillan. The world's first computer password? it was useless too. 2012. https://www.wired.com/2012/01/computer-password/.

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

[PJ16] 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.

[QQQ$^+$89] 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.

[SAH$^+$08] Dan Simon, Bernard Aboba, Ryan Hurst, et al. The eap-tls authentication protocol. *RFC 5216*, 2008.

[Sal04] Joseph Salowey. Extensible authentication protocol (eap) registry, method types. *IANA Extensible Authentication Protocol (EAP) Registry*, 2004.

[Shi07] Robert Shirey. Internet security glossary, version 2. Technical report, RFC 4949, August, 2007.

[Sim94] William Simpson. *RFC1661: the point-to-point protocol (PPP)*. RFC Editor, 1994.

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