Zero-Knowledge Authentication

Jakob Povšič
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Glagoljaška 8, 6000 Koper, Slovenia
jakob.povsic@gmail.com

Andrej Brodnik
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Glagoljaška 8, 6000 Koper, Slovenia
andrej.brodnik@upr.si

Abstract

Zero-Knowledge proofs (ZKPs) enable proving of mathematical statements, revealing nothing but their validity. We design an authentication system with a ZKP as a password verification mechanism within the Extensible Authentication Protocol (EAP) framework. Designing a secure password authentication system requires us to adopt security practices for protecting ourselves against the vulnerabilities of passwords. Integrating said practices is not trivial because of the tight coupling with the password verification method.

Keywords extensible authentication protocol, zero-knowledge proofs, authentication, cryptography, key-stretching, passwords, quadratic residuosity problem

1 Introduction

Today privacy is a necessary sacrifice we have to make in order to take part in the digital world, imperative to our modern life. Every day, more digital systems gain access to our personal information. While this practice is often a necessary evil, many companies seek to exploit this position. Zero-knowledge proofs (ZKPs) are an intriguing cryptographic phenomenon for proving mathematical statements without revealing *why* they are true, and have the potential to change how our data exists in the digital space.

Our focus will be to define a simple use for zero-knowledge proofs. We will design an authentication system using a zero-knowledge proof as a password verification method, as an authentication method in the extensible authentication protocol (EAP). When designing the system, we need to protect ourselves from vulnerabilities of passwords. However, integrating security methods presents a challenge because of the zero-knowledge proof system.

2 Extensible Authentication Protocol

Extensible authentication protocol [10] (EAP) is a general purpose authentication framework designed for network access authentication. EAP defines a set of messages that support negotiation and execution of a variety of authentication protocols. EAP is a two-party protocol between a *peer* and an *authenticator* at each end of a link.

Messages. The peer and the authenticator communicate by exchanging *EAP messages*. The protocol starts with the authenticator sending a message to the peer. They keep exchanging messages until the authenticator can either authenticate the peer or not. Messages are exchanged in a lock-step manner, where an authenticator sends a message and the peer responds to it. The authenticator dictates the order of messages, meaning it can send a message at any point of communication, as opposed to the peer, which can only respond to messages from the authenticator.

Messages are composed of fields, each field length is multiple of an octet of bits (Table 1). We will store our authentication method data within the *Type-Data* field.

Our EAP method is identified by the *Type* 84.

3 Zero-Knowledge Proofs

Zero-Knowledge Proofs [5, 6, 7] (ZKPs) are a concept in cryptography for proving the validity of mathematical statements. What makes them particularly interesting is that ZKPs can prove a statement revealing no information about why a statement is true, hence the term *zero-knowledge*. In mathematics, theorem proofs are logical arguments that establish truth through inference rules of a deductive system based on axioms and other proven theorems. ZKPs are probabilistic, meaning they *convince* the verifier of the validity. We use the term convince, because ZKPs are not absolute truth, but the probability of someone being convinced by a false statement is arbitrarily small.

3.1 ZKP System for the Quadratic Residuosity Problem

Definition 3.1 (Quadratic Residuosity Problem) *Given an integer x, a semiprime modulus $n = pq$, where p and q are unknown different primes, and a Jacobi symbol value $\left(\frac{x}{n}\right) = 1$. Determine if x is a quadratic residue modulo n or not.*

The *law of quadratic reciprocity* enables efficient computation of the Jacobi symbol value $\left(\frac{x}{n}\right)$. However, when $\left(\frac{x}{n}\right) = 1$, it does not tell if x is a quadratic residue modulo n or not. x is only a quadratic residue if it's a quadratic residue of both modulo p and q ($\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$). To compute this, we would have to know the factorization of n. However, since n is a product of two primes $pq = n$, this is computationally hard [2]. The only efficient way

DOI https://doi.org/10.18690/978-961-286-516-0.2
ISBN 978-961-286-516-0

Table 1: EAP Message Format

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

to prove x is a quadratic residue modulo n, is with the root w. The problem acts as a *trapdoor* function, where it's hard to prove if x is a quadratic residue modulo n solely from x and n, while it is easy to prove when you know its root w.

Authors [7] described a ZKP system for the *quadratic residuosity problem*. To prove x is a quadratic residue modulo n in zero-knowledge we need to prove the existence of the root w, where $w^2 \equiv x \pmod{n}$, without revealing w to the verifier.

n Semiprime, where Jacobi $\left(\frac{x}{n}\right) = 1$
x Residue, where $w^2 \equiv x \pmod{n}$
w Root

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

Table 2: ZKP Authentication with EAP

In Table 2 of the ZKP authentication process, the middle spaces represent the EAP message *Type-Data* field.

The prover begins by picking a random integer u from the field $\mathbb{Z}_n$, computing $y = u^2 \pmod{n}$, and sending y to the verifier. The verifier picks a random bit b and sends it to the prover. The prover computes the value z with b and sends it back. The verifier checks the proof by asserting $z^2 \equiv yx^b \pmod{n}$, this is possible since

$$z^2 \equiv yx^b \pmod{n}$$
$$(uw^b)^2 \equiv u^2(w^2)^b \pmod{n}$$
$$u^2w^{2b} \equiv u^2w^{2b} \pmod{n}.$$

For each round a cheating prover has a $\frac{1}{2}$ probability of succeeding by correctly guessing the value of the random bit b. To improve the strength of the proof, we repeat this process m times for a confidence of $1 - 2^{-m}$.

To use this protocol as a password verification method, we can treat the root w as the password $p = w$ known by the peer. The ZKP protocol proves that x is a quadratic residue modulo n, by proving the knowledge of the root w, where $w^2 \equiv x \pmod{n}$. The peer will prove that x is a quadratic residue modulo n, to do this however, the peer needs to prove the knowledge of the password $p = w$. With this, the authenticator can assert that the password is valid.

4 Password Protection

Password cracking [1] is an *offline attack* [8], where an attacker extracts passwords from data used by the authentication system for password verification. Protecting passwords on the data layer is of critical importance. *Key-stretching*, [9, 1] also called *password hashing*, is the industry standard method for improving security of low entropy secrets like passwords.

The quadratic residue x is derived from root $w = p$ and persistently stored with the authenticator. This introduces a vulnerability, as an attacker with access to x could crack the password w in an offline attack. To provide adequate security, we need to use key-stretching in our authentication method. A common application of a key-stretching method is to transform the vulnerable data stored in the authentication system. However, this approach doesn't work in our case. Let us revisit how the authenticator verifies the proof, and why key-stretching the password verification data (x) data is an issue. We'll begin by assuming the system can verify the proof and key-stretching the password verification data (x) data. As we define our process, we will see why it is not possible.

Key-Stretching x. On the last step of the protocol the authenticator verifies that

$$z^2 \equiv yx^b \pmod{n}.$$

If we stretch x with a function H and a salt s

$$H(x, s) = x_H,$$

we can then verify the proof with an inverse function H^{-1}

$$z^2 \equiv yH^{-1}(x_H, s)^b.$$

This is possible assuming a polynomial algorithm H^{-1} exists, however, since key-stretching methods are based on hashing functions (one-way functions), we know that the probability of a polynomial algorithm H^{-1} to successfully compute a *pseudo-inverse* is negligibly small. For all positive integers c [4]

$$\Pr[H(H^{-1}(H(x))) = H(x)] < |x|^{-c}.$$

Even if given unbounded time and resources, the *pseudo-inverse* $x' = H^{-1}(H(x))$ might not be equal to $x' \neq x$. The set $x, x' \in I_x$ are all values that map into $H(x) = H(x')$, and since H is not injective we know that $|I_x| \geq 1$. Meaning that the probability that $x' = x$ is

$$\Pr[H^{-1}(H(x)) = x] = \frac{1}{|I_x|}.$$

Key-stretching x prevents us from verifying the ZKP. However, by increasing the entropy of the root w, we can eliminate the vulnerability and ensure adequate security. Our new approach won't treat the password $p \neq w$ as the

root w. However, we will use the password p to derive the root $w = H(p, s)$, using a key-stretching function H and salt s. This way we've ensured the same level of protection against offline attacks as if we stretched the data stored in the system. And because we didn't transform x, we can verify the proof without being affected by issues mentioned in the previous paragraph. A similar approach is used in the PPP EAP SRP-SHA1 protocol [3]. Earlier we argued the ZKP works as a password verification method because $p = w$, this argument isn't true anymore. However, even though $w \neq p$, the peer can only derive w knowing the password p, so when the peer proves the knowledge of w, it can only be so because they know p as well.

5 Secure Authentication

The authentication process now begins with the peer sending his identifier to the authenticator and the authenticator responding with the peer's unique salt s and modulo n. The peer can now derive the root w from the password p and salt s. This part of the process (Steps 1. and 2. in the Table 3) happens only once.

The peer can then authenticate by following the process as described in §3.1. This part (Steps 3., 4. and 5.) of the process is repeated m times for a confidence of $1 - 2^{-m}$.

The middle space in the Table 3 represents the *Type-Data* field of the EAP messages.

	Peer		Authenticator
1		$\xrightarrow{I}$	
2	$w = H(p, s)$	$\xleftarrow{s,n}$	
3	$u \quad_R \mathbb{Z}_n$		
	$y = u^2 \pmod n$	$\xrightarrow{y}$	
4		$\xleftarrow{b}$	$b \quad_R \{0, 1\}$
5	$z = uw^b \pmod n$	$\xrightarrow{z}$	verify $z^2 \equiv yx^b \pmod n$

Table 3: Improved ZKP Authentication with EAP

Let us examine the EAP messages (Figure 1) of the authentication process described in Table 3. The mapping between EAP messages and the steps in Table 3 is *not one-to-one*. We merged some steps to reduce the number of message exchanges required for the process to complete.

Identity This message is used to query the identity of the peer. In a system with multiple peers, this is required to identify the peer authenticating, and to find the correct salt s and quadratic residue x. (Table 3, Step 1.)

Setup The peer needs both the salt s and the modulus n to compute the proof, however, he only knows the password p. Once the peer identifies himself, the authenticator needs to send him the salt s and modulus n in the *setup request* message. (Table 3, Step 2. and 3.)

Verification With this message pair the peer and the authenticator exchange data to compute and verify the proof. The authenticator sends the random bit b and the peer responds with the proof z. The peer also sends the y_{i+1} for the next verification round $i + 1$, this is done as an optimisation to improve the speed of the process. (Table 3, Step 4., 5. and 3.)

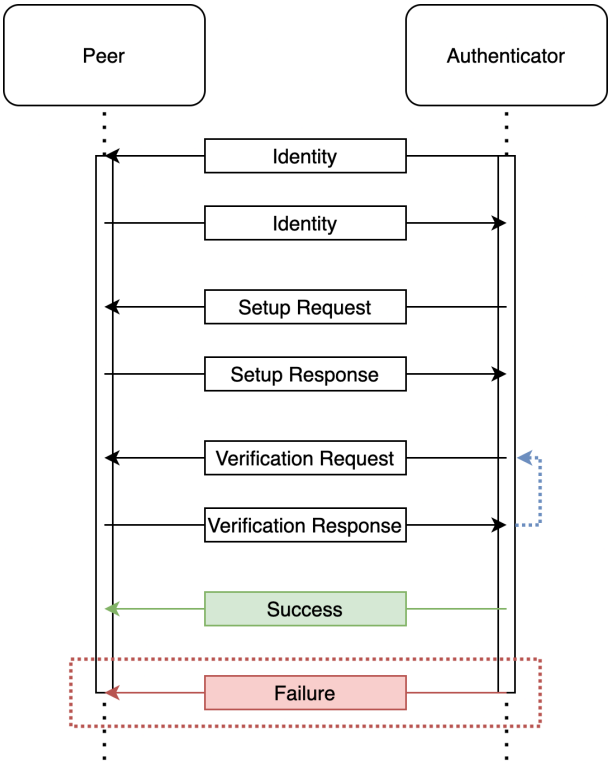

Figure 1: EAP Method Execution

Success/Failure After each *verification* message, the authenticator verifies the proof, and once it's done successfully for m rounds, the authenticator sends the *success* message. However, if the proof isn't valid, the authenticator must send a *failure* message.

6 Conclusions and Future Work

The aim of this work was to study the utility of zero-knowledge proofs as an EAP authentication method. We've presented an EAP method using a ZKP system for password verification. Additionally, we ensured adequate password protection by using a key-stretching method.

We have been successful in our goal of studying and using the ZKP protocol. While theoretically interesting the system's performance may not appropriate for real-world applications. The iterative nature of the underlying ZKP protocol accumulates communication latencies, slowing down the system.

Future work.

- The EAP method presented in this work can be implemented and tested in a real-world environment.

- The ZKP protocol used in this work is a first generation protocol. Today there are many newer protocols that have solved many shortcomings of the older generation ZKPs. Using a newer generation ZKP protocol can improve the performance of the authentication system.
- The ZKP protocol we've examined is iterative, which can cause worse performance. A parallel ZKP construction is assumed to have a weaker strength of zero-knowledge. However, in a real-world application, the performance improvements might justify the theoretical shortcomings.

References

[1] Blocki, J., Harsha, B., and Zhou, S. On the economics of offline password cracking. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), IEEE, pp. 853–871.

[2] Buchmann, J. A. *Factoring*. Springer US, New York, NY, 2001, pp. 171–183.

[3] Carlson, J. D., Aboba, D. B. D., and Haverinen, H. PPP EAP SRP-SHA1 Authentication Protocol. Internet-Draft draft-ietf-pppext-eap-srp-03, Internet Engineering Task Force, July 2001. Work in Progress.

[4] Goldreich, O. *Foundations of cryptography: Volume 1, basic tools*. Cambridge university press, 2007.

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

[6] Goldreich, O., Micali, S., and Wigderson, A. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. vol. 263, pp. 171–185.

[7] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM Journal on computing 18*, 1 (1989), 186–208.

[8] Grassi, P. A., Garcia, M. E., and Fenton, J. L. NIST Special Publication 800-63-3 Digital Identity Guidelines. *National Institute of Standards and Technology, Los Altos, CA* (2017).

[9] Hornby, T. Salted password hashing-doing it right, 2016.

[10] Vollbrecht, J., Carlson, J. D., Blunk, L., Aboba, D. B. D., and Levkowetz, H. Extensible Authentication Protocol (EAP). RFC 3748, June 2004.