PROVE YOURSELF

Zero-Knowledge Password Authentication

Johnathon Schultz

APRIL 25, 2016
FURMAN UNIVERSITY
Department of Computer Science

Abstract

Your information is online. Secure? You are probably more succeptable than you believe. A large trust is placed on Web applications to securely store user information due to the prevalence of password-based authentication and current authentication protocols throughout the Web. The servers where this sensitive information resides are vulnerable. Zero-Knowledge Password Authentication offers the ability to discern and verify genuine users without transmission or storage of users' passwords. But, time is a factor. The the amount of time required to obtain confident authentication of a user in the original implementation of Zero-Knowledge Password Authentication failed to keep the user's flow uniterrupted. This paper proposes a protocol to reduce communication time by concatenating encrypted Zero-Knowledge Password Authentication challenges in an effort to minimize network overhead. With optimizations, Zero-Knowledge Password Authentication is on its way to rival conventional authentication protocol methodologies.

Introduction

Everyone logs in. Modern day lives necessitate passwords. Most of the time, however, the security of the system beyond the front page is never really a forethought. What happens once your username and password leave your machine? In a standard password authentication model, some form of each password is stored on the sever in the back end so that the password can be compared for future logins. Servers are

vulnerable. But what happens if the server is compromised or malicious? Threat levels range signficantly, as seen in Figure 1.

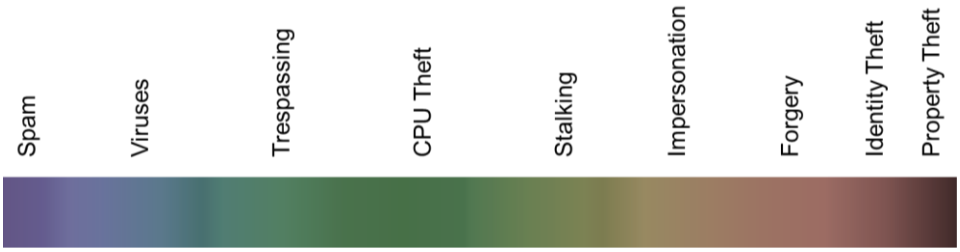

Figure 1

Although most of the threats to a user are seemingly harmless, given proper motivation, malicous hackers are given the resources to greatly affect the security and lives of individuals. Access to the most sensitive account details can be prevented simply by not storing account passwords on the server. Hackers can't steal what isn't stored.

In an Internet driven society, authentication (ensuring an individual is who they claim to be) is a major conundrum. Providing the ability to discern and verify genuine users amid mistyped passwords and hackers, while protecting their security and trust from data breaches, appears seemingly insurmountable[1]. Does authentication require the transmission and storage of users' passwords? What if a user's identity could be verified within a statistically insignificant margin of error? Zero-Knowledge Password Authentication (ZKPA) provides a method for verifying users with a server without revealing or passing along any information regarding the users other than the fact that the users know their password.

[1] "Passwords," "Threats to Passwords."

Requirements

By definition, ZKPA must satisfy three properties: completeness, soundness and zero-knowledge. To statisfy completeness, a verifier must be convinced by an honest client possessing the password. In other words, a correct authentication attempt must result in successful authentication of a genuine user. Enforcing that an honest verifier will not be convinced by a malicious client not possessing the password (aside from a statistically insignificant probability) satisfies the property of soundness. Put another way, an legitimate server will not authenticate a malicious client attempting to simulate, hack or otherwise circumvent the password authentication. Finally, to satisfy zero-knowledge, a malicious verifier must not learn anything from an honest client possessing the password other than the fact that they possess it. Meaning, upon attempted authentication with a password, a malicious server will not learn anything from a genuine user other than the fact they possess the password[2,3,4].

Implementation

As with most cryptographic algorithms, ZKPA relies on computational assumptions of cyclical groups. Prime order groups are typically implemented since groups of prime order are cyclic. One possible implementation takes advantage of relative primes invertable through modulo multiplication. Further, composite order groups, comprised of two distinct primes, reveals cardinality through modulo muliplication. A final possible implementation enacts an ellipical curve over a prime-field with similar properties of

[2] "Zero-knowledge Proof," “Definition.”
[3] Kiefer, "Advancements in Password-based Cryptography,” 37.
[4] Kurmi and Sodhi, "A Survey of Zero-Knowledge," 494-500.

inversion from an abelian group[5]. Each method has it's own advantages and disadvantages, however, for simplicity, we will use an implementation of composite order groups through discrete logarithm.

The interface of the proposed system is familiar and identical to that of which is currently widely accepted. A user logs in with a username and password on a form as seen in Figure 2.

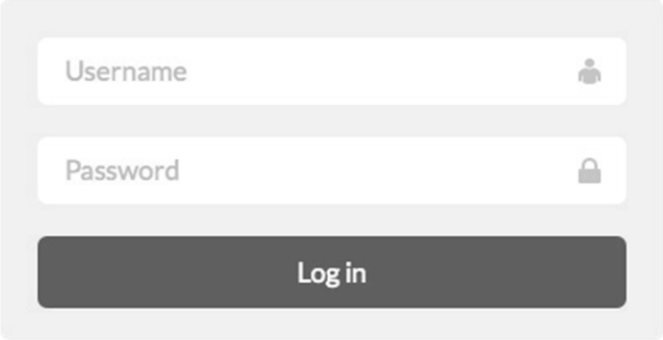

Figure 2

Maintaining the same username and password model opens the door for current content providers to implement such a system with limited impact on both the user and the content provider. User's will be able to login as they normally would–by entering their username and password in the form and clicking login. However, the interface is where the similarity ends.

On the other end, content providers would not store passwords but instead a distributed key value that is a function of the password. Doing so, the calculation for

[5] Kiefer, "Advancements in Password-based Cryptography," 21-23.

authentication reveal nothing but random numbers. Even having access to the database does not gleam any information about a user's password.

When a user wishes to attempt to login, the user enters their username and password on their machine. The following execution of the ZKPA algorithm takes place:

Given a value *y*, a large prime *p* and a generator *g*, such that $y = g^x \mod p$, assuming the value of *y* is already distributed to the verifier, such that at a later time, proving knowledge of *x* is equivalent to proving identity of the user.

1. The user's machine calculates the hash of their password, *x*.
2. For each iteration
    a. The user's machine generates a random number *r*, computes *C*, such that $C = g^r \mod p$, and discloses *C* to the server.
    b. The server then issues one of the following two challenges:
        i. The server requests the value of *r* from the user.
        ii. Or, the value of $(x + r) \mod (p - 1)$.
    c. The user's machine generates the response to the challenge for the server.
    d. The server verifies the response:
        i. If *r* was requested, the server can compute $g^r \mod p$ and verify that it matches *C*.
        ii. If $(x + r) \mod (p - 1)$ was requested, the server can verify that *C* is consistent with this, by computing $g^{(x+r) \mod (p-1)} \mod p$ and verifying that it matches $C \cdot y \mod p$.

Therefore, “by executing a large enough number of interations, the probability of a malicious client succeeding in false authentication can be made arbitrarily low[6,7,8].”

Probablitiy

In any particular single instance, a malicious user has a 50% probability of guessing the challenge correctly[9], however, with each additional iteration, the overall probability of simply guessing all correct challenges becomes increasingly small, as seen in Figure 3 below. The probability of correctly guessing the challenges decreases exponentially based on a function of the number of iterations completed ($\frac{1}{2^n}$, where *n* is the number of iterations).

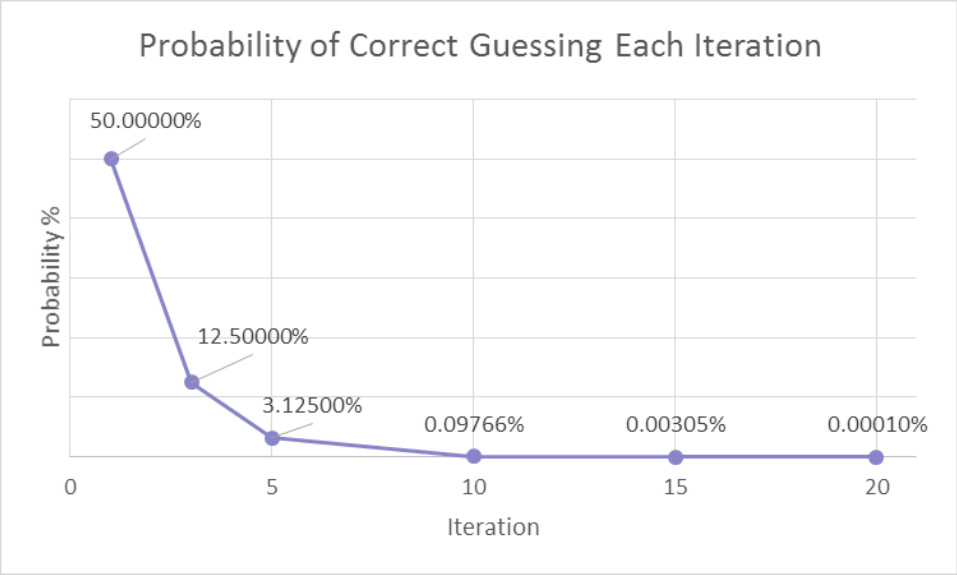

Figure 3

[6] "Zero-knowledge Proof," “Practical Examples.”
[7] Nyguyen, Rudoy, and Srinivasan, "Two Factor Zero Knowledge Proof," 2-4.
[8] Kurmi and Sodhi, "A Survey of Zero-Knowledge," 494-500.
[9] "Zero-knowledge Proof," “Practical Examples.”

If the server issues a challenge to the malicous user other than the one they were expecting, the complexity of formulating the expected response without the password is on the order of solving the discrete logarithm problem. Therefore, if the user fails to complete the requested challenges as expected, the server can be confident the user does not have the password. To the contrary, if a user repeatedly completes the requested challenges as expected for enough iterations, the server can conclude with great certainty that the user possesses the password within a statistically insignificant margin of error.

It is important to note that the implementation this paper chose is simplified. More complex algorithms produce a more drastic fall-off on the probability curve. To reduce the probability of a malicous user guessing the requested challenges as expected for each iteration, an algorithm must be formed that expects more than two possible responses to the challenges. Doing so effectively increases the complexity to a function of $\frac{1}{P^n}$, where *P* is the number of possible responses to a challenge and *n* is the number of iterations. An example of an implementation with higher single-iteration complexity is the Modification of Diffie-Hellman Key Exchange Algorithm for Zero-Knowledge Protocol[10].

Another important analysis is the threat consideration of standard password authentication protocols. Given any arbitrary username and password on the Internet, the probability that they are compromised is not zero. Standard password authentication protocols can verify a password's correctness with absolute certainty, whereas ZKPA can only become confident in the password within a statistically insignificant margin of error.

[10] Kurmi and Sodhi, "A Survey of Zero-Knowledge," 497-499.

However, the amount of time that the password is exposed to threat in standard password authentication protocols is much greater, leaving them more suceptible. In the implementation of ZKPA, by not having any form of the password saved on the server, the avenue of attack is greatly reduced. Secondly, the amount of time for a given attempt of ZKPA is not instantaneous; therefore, an attack on the avaliable front, say for example bruteforce, would take significantly longer than a standard password authentication protocol. Nonetheless, simple password attempt limiting techniques could easily mitigate such attacks. In the end, ZKPA proves to be less suceptible to password related threats.

Performance

“An important challenge for [ZKPA is] to satisfy the existing response-time standards. Such a standard was proposed in 1968 by Miller[11]. He defined three main time constraints: 0.1 second is for keeping the user attention attracted; 1 second is for keeping a user flow though uninterrupted; Finally, 10 seconds for keeping user's attention on the dialog. Three decades after his findings, those rules are still applicable…[12]”

The main drawback of ZKPA is in fact performance. In implementation, ZKPA failed to keep the user’s flow uniterrupted; and, in some circumstances, even failed to meet the time requirement to keep the user’s attention. Each ZKPA iteration took aproximately 400ms on a desktop machine and 600ms on a mobile device (Figure 4). The response time for an individual iteration ranged from 57ms to 1230ms on desktop and 356ms to1860ms on mobile (see Appendix). Since similar computations occurend in

[11] Miller. Response time.
[12] Grzonkowski, Zaremba, Zaremba, and Mcdaniel. "Extending Web Applications." 4.

each iteration, the variance in timings for each iteration can be accounted largely to network overhead.

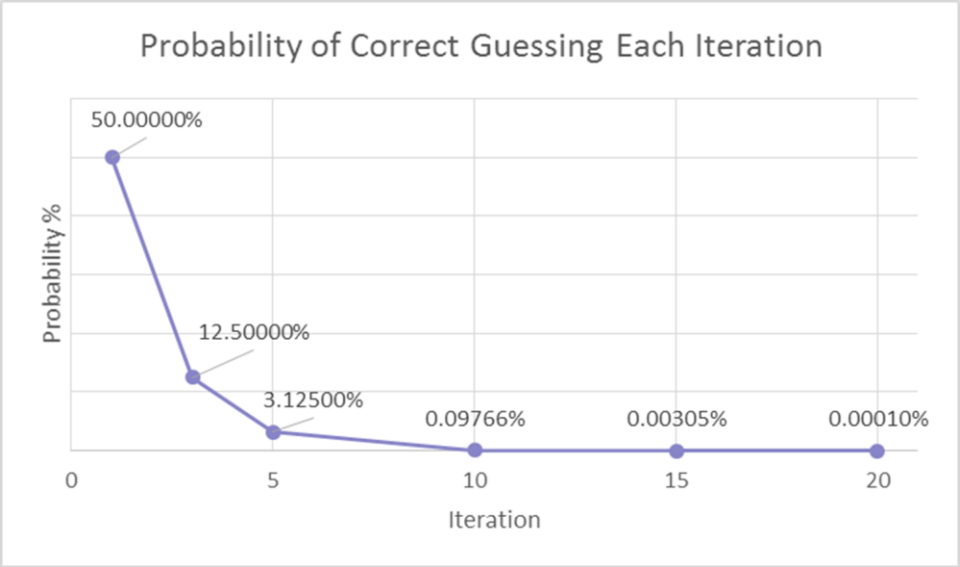

Figure 4

Altogether, the combined iteration time required to confidently verify a genuine user challenged the user's attention even in the best case. ZKPA took 8.5 seconds to complete on a desktop machine and 12.8 seconds on a mobile device (Figure 5).

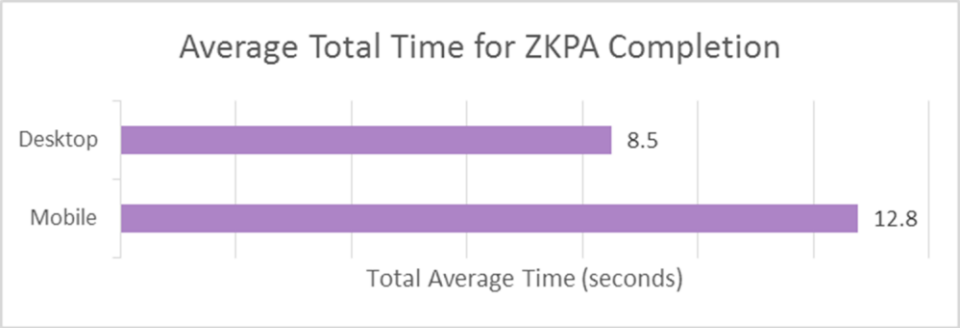

Figure 5

In order to obtain a 1 second response time, the algorithm would need to see a 90 – 95% reduction in time required for confident authentication. Improvement of that magnitude is not likely; however, the proprosed additions that follow will reduce the amount of network

communication trips down from two per iteration (for a total of 40 over 20 iterations) down to a total of twp trips from client to server with a larger concatenated challenge request.

Additions

In an effort to reduce network overhead while still statifying the requirements of ZKPA, each challenge must be calculated independently and answered correctly prior to receiving the next challenge. By employing a Triple-DES block cipher to encrypt the random iteration challenges with cipher block chaining, such requirements will be satisfied. In turn, by sending all of the challenges at once, the network overhead is reduced to two trips independent of the number of iterations challenges sent.

As shown in Figure 6, the implementation will function largely the same. The user will first enter their username and password into the browser. The server receives the request and generates the random iteration challenges. However, instead of generating the iterations one at a time, the sever generates all of the iteration challenges at once and concatenates them together in an encrypted block. By employing cipher block chaining, the initialization vector for each challenge will be different; however, by basing the initialization vector on the correct response to each subsequently previous challenge, the server can ensure the user’s machine can not gleam any useful information out of future challenges. Therefore, the user’s machine’s response for a given iteration is required to be correct for the user to decrypt the next challenge correctly. If any response is incorrect, subsequent challenges will be misinturpreted upon decryption and the user’s machine will essentially be providing a response to the wrong question. Once the responses have been formulated for each challenge by the user’s machine, the response is sent back to the server for verification. If all of the responses are correct, the user is authenticated and

the server can be confident within a statistically insignificant margin of error that the user does in fact have the password. To the contrary, if any one response is incorrect, the server can be entirely confident the user does not possess the password.

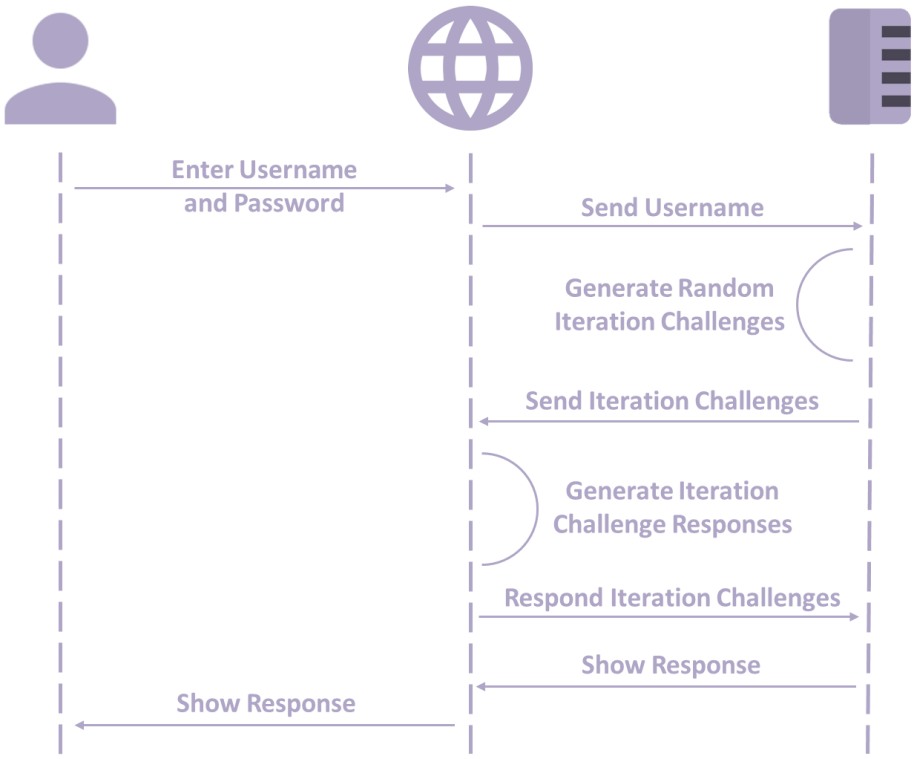

Figure 6

Conclusion

In cosidering security beyond the login form, how users' paswords are handled matters. Everyone logs in; therefore, passwords are a necessity in most people's lives. In a standard password authentication model, some form of each password is stored on the sever in the back end so that the password can be compared for future logins. Doing so leaves open the possibility for malicous activity in obtaining their password through common server attacks. Access to the most sensitive account details can be prevented simply by not storing account passwords on the server. By not storing the information, it

is then not avaliable to be obtained by a malicious individual by means of hacking the server.

Due to the prevalence of password-based authentication and current authentication protocols throughout the Web, as it stands, a large trust is placed on Web applications to securely store user information. ZKPA offers the ability to discern and verify genuine users without transmission or storage of users' passwords. However, the limiting factor for ZKPA is the amount of time required to obtain confident authentication of a user. This paper proposed a protocol to reduce communication time by concatenating encrypted challenges. Future work in reducing computation and communication costs in an effort to minimize average total time for ZKPA will allow ZKPA to rival conventional authentication protocol methodologies.

References

Bellovin, S.m., and M. Merritt. "Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks." *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy* (n.d.): n. pag. Web.

Grzonkowski, Sławomir, Wojciech Zaremba, Maciej Zaremba, and Bill Mcdaniel. "Extending Web Applications with a Lightweight Zero Knowledge Proof Authentication." *Proceedings of the 5th International Conference on Soft Computing as Transdisciplinary Science and Technology - CSTST '08* (2008): n. pag. Web.

Kiefer, Franziskus. "Advancements in Password-based Cryptography." (n.d.): n. pag. University of Surrey, Jan. 2016. Web.

Kurmi, Jitendra, and Ankur Sodhi. "A Survey of Zero-Knowledge Proof for Authentication." *International Journal of Advanced Research in Computer Science and Software Engineering* 5.1 (2015): 494-501. Web.

Miller, R. B. Response time in man-computer conversational transactions. In Proc. AFIPS Fall Joint Computer Conference Vol. 33, pages 267-277, San Francisco, Calif, 1968.

Nyguyen, Quan, Mikhail Rudoy, and Arjun Srinivasan. "Two Factor Zero Knowledge Proof Authentication System." (n.d.): n. pag. 2014. Web.

"Passwords: Threats and Counter-Measures." *Jisc Community*. Janet CSIRT, n.d. Web. 28 Mar. 2016.

"Standard Specifications for Public-Key Cryptography." *IEEE* (n.d.): P1363.2. Web.

"Zero-knowledge Proof." *Wikipedia*. Wikimedia Foundation, 27 Apr. 2016. Web.

Appendix

Desktop iteration times across five trials:

DESKTOP	1	2	3	4	5	AVG
Init	106	117	177	120	150	134
1	388	663	837	682	170	548
2	171	357	166	598	657	389.8
3	92	57	754	674	362	387.8
4	400	145	108	571	428	330.4
5	321	369	592	126	479	377.4
6	401	661	537	642	788	605.8
7	212	465	61	769	516	404.6
8	688	209	617	137	403	410.8
9	694	90	623	661	277	469
10	585	682	65	482	267	416.2
11	1230	470	152	705	659	643.2
12	584	296	139	149	557	345
13	392	507	125	575	449	409.6
14	107	695	277	327	465	374.2
15	139	66	101	702	93	220.2
16	351	299	396	356	192	318.8
17	811	570	195	531	898	601
18	747	196	111	324	836	442.8
19	105	124	227	435	374	253
20	598	461	322	697	59	427.4
SUM	9.122	7.499	6.582	10.263	9.079	8.509

Mobile iteration times across five trials:

MOBILE	1	2	3	4	5	AVG
Init	411	588	583	413	447	488.4
1	554	506	537	511	453	512.2
2	660	427	412	724	731	590.8
3	600	729	447	648	547	594.2
4	1540	403	405	842	551	748.2
5	361	454	449	502	467	446.6
6	464	878	451	450	640	576.6
7	458	700	546	455	818	595.4
8	460	930	375	452	377	518.8
9	460	495	469	871	721	603.2
10	581	698	449	739	423	578
11	923	645	759	498	446	654.2
12	930	600	411	657	464	612.4
13	1410	631	456	455	451	680.6
14	1860	744	446	452	447	789.8
15	1120	356	604	559	728	673.4
16	929	601	540	371	424	573
17	581	564	836	473	509	592.6
18	843	489	553	686	640	642.2
19	554	788	835	561	687	685
20	872	559	565	361	728	617
SUM	16.571	12.785	11.128	11.68	11.699	12.7726

Desktop iteration min, max, average, and standard deviation:

DESKTOP	
Min	57
Max	1230
Average	405.19
Std Dev	247.35

Mobile iteration min, max, average, and standard deviation:

MOBILE	
Min	356
Max	1860
Average	608.22
Std Dev	235.68