Implementing Zero-Knowledge Authentication with Zero Knowledge (ZKA_wzk)

Lum Jia Jun, Brandon
Temasek Polytechnic
lumjjb@gmail.com

Abstract

A practical web/python implementation of Zero-Knowledge Authentication protocol without any prior knowledge of the concept of Zero-Knowledge Proof.

The Zero-Knowledge Proof is a concept used in many cryptography systems. It allows a party to prove that he/she knows something (i.e. credential), without having to send over the value of the credential. In this implementation, it will be used to prove the password of the user without sending over the actual password. The system also allows for no password hashes to be stored on the server.

The purpose of the implementation is to make implementing the Zero-Knowledge Proof Authentication portable and easily customizable. This is achieved by using python based scripts in web applications to simulate the protocol.

1. Introduction

This section will cover the various background knowledge and reasons relating to the paper. This covers concepts such as Zero-Knowledge Proof and several current configurations of corporate sites.

1.1. What is Zero-Knowledge Proof

Zero-Knowledge proof is a much popular concept utilized in many cryptography systems. In this concept, 2 parties are involved, the prover A and the verifier B. Using this technique, it allows prover A to show that he has a credential (for example, a credit card number), without having to give B the exact number. The reason for the use of a Zero-Knowledge Proof in this situation for an authentication system is because it has the following properties:

- **Completeness**: if the statement is true, the honest verifier (that is, one following the protocol properly) will be able to prove that the statement is true to an honest verifier everytime.
- **Soundness**: if the statement is false, it is not possible (with a very small chance) to fake the result to the verifier that the statement is true.

- **Zero-knowledge**: if the statement is true, the verifier will not know anything other than that the statement is true. Information about the details of the statement will not be revealed.

For a more detailed explanation of the concept of Zero-Knowledge Proof, you may refer to the paper titled "How to Explain Zero-Knowledge Protocols to Your Children" (Quisquater and others, 1990)[1].

1.2. Rising use of web applications

There has been an increase of usage of web-based applications as compared to 10 years ago. This is mainly due to several advantages of web applications that they provide over traditional software [based on the article (Yadav 2008)[2]]:

- Compatibility with devices, web browsers & operating systems
- Capability of performing tasks which were done using host-based software
- Development of new web technologies, languages and procedures has given way to the development of new dynamic applications (i.e. AJAX)
- Easy to use and are more presentable and attractive
- Does not require any additional hardware or software (installation, etc.) configuration
- Data security is there as it is stored in one central server and not individual computers
- A custom build web application certainly costs less than the off the shelf applications and provides greater efficiency and reduced maintenance

That given, it is important for us to focus on how to improve the security of web-based applications, and maintain the ease of use to develop and implement them.

1.3. Security issues relating to web applications

There are many vulnerabilities and attack vectors for web-based application. This includes both web-specific (i.e. Cross-Site Scripting), as well as generic (i.e. Password Sniffing), all of which leave the user susceptible to being victims of identity theft.

"Almost 80 percent of more than 3,000 software security flaws publicly reported so far this year have been in Web technologies such as Web servers, applications, plugins, and Web browsers. " - According to a report (Vijayan 2009) by application security vendor Cenzic[3]

Also, relying on service based security solutions such as SSL may fully protect an application due to the introduction of new web concepts such as cloud computing and the growing complexity of attack vectors.

[1] "How to Explain Zero-Knowledge Protocols to Your Children." *Advances in Cyptology - CRYPTO '89.* Jean-Jacques Quisquater, Louis C. Guilou, Thomas A. Berson, 1990. 628-631.

[2] Yadav, Sujeet. *Rising Popularity of Web Application Development*. August 28, 2008. http://ezinearticles.com/?Rising-Popularity-of-Web-Application-Development&id=1449765

[3] Vijayan, Jaikumar. *Web application security is growing problem for enterprises.* November 11, 2009. http://www.infoworld.com/d/security-central/web-application-security-growing-problem-enterprises-843

"Web security, application security, software security, or whatever you want to call it will soon come into its own. It will no longer be acceptable, feasible, or even seriously suggestible to run for cover by simply adding more firewalls and SSL. Things like "the cloud" will help make this fail plain as day." - Jeremiah Grossman (2010)[4]

1.4. Current web application login process

The most common login system used in web application currently is through the use of a form submission of a username and passwords enabled with SSL communication. In more secure systems, the password is hashed using a javascript-based md5 hash before sending it over.

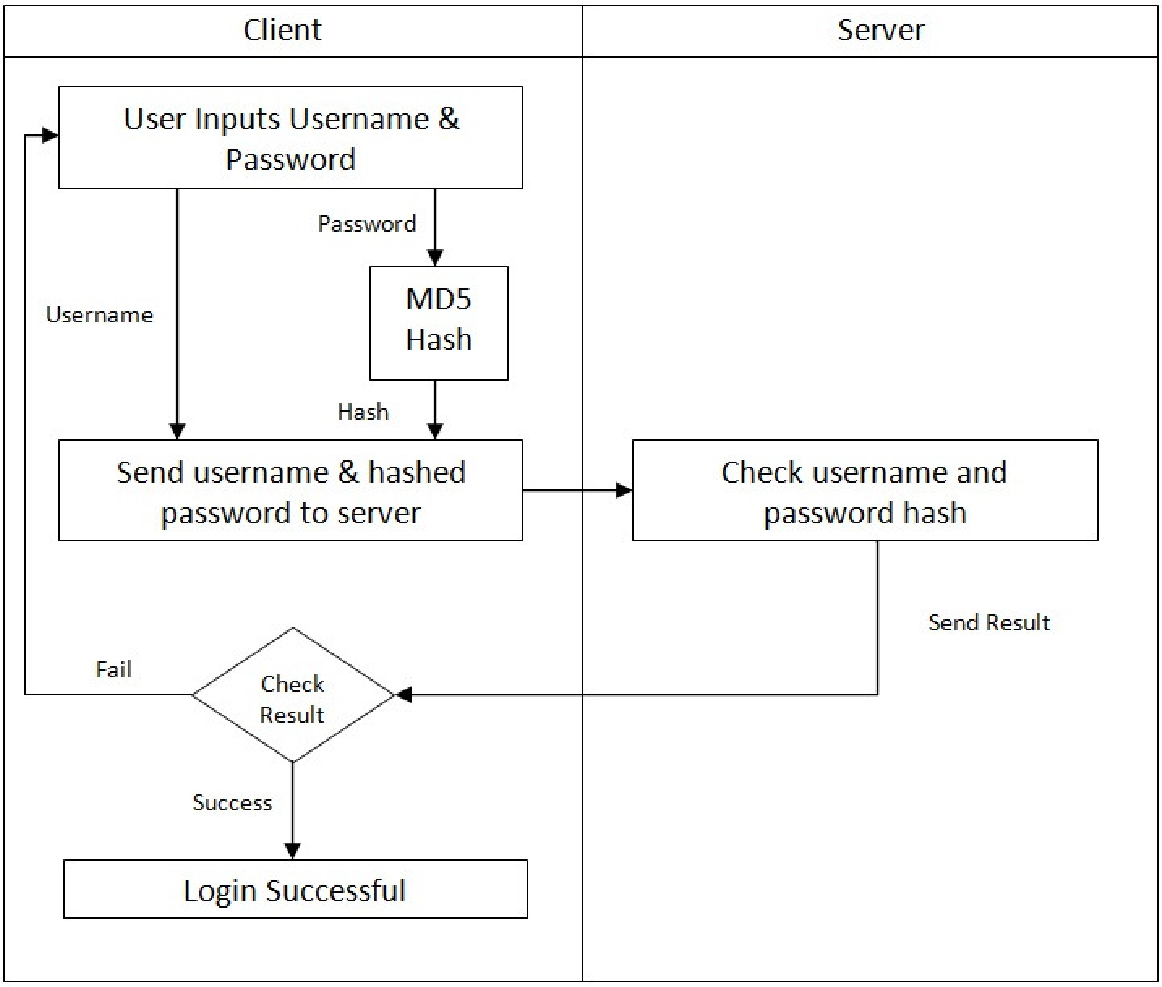
Client
Server
User Inputs Username & Password
Password
Username
MD5 Hash
Hash
Send username & hashed password to server
Check username and password hash
Send Result
Fail
Check Result
Success
Login Successful

Fig. 1.1. Traditional Authentication System

[4] Grossman, Jeremiah. *Be Ready -- With Answers.* February 2, 2010. http://jeremiahgrossman.blogspot.com/2010/02/be-ready-with-answers.html

However, by using this traditional authentication systems, there are many problems which will arise such as sniffing and hash attacks. These problems will be described in the Section 2: Problems to Solve.

2. Problems to solve

This section covers the various problems which are prevalent that leads to the implementation of the ZKA_wzk.

2.1. Problem of the usage of unsecure wireless networks

Today, the usage of wireless hotspots is growing in popularity. Though these services provide convenience and ease of access, they are, more often or not, unsecure in terms of transmission.

"I think a lot of people don't realise that using public Wi-Fi that's insecure is pretty much like writing your bank details onto a postcard and popping it in the post and being surprised that someone's read it." - Tom Illube (2009), chief executive of internet security firm Garlik[5]

Therefore, users may be susceptible to password sniffing and may end up not realizing the ‘full potential’ of these services. The existent of non-encrypted transmission of data through these mediums result in the existence of several problems such as password sniffing. Please refer to Section 2.2: Problem of sending over password hashes.

2.2. Problem of 3G Networks

Another alternative which many people rely on today is 3G, broadband on mobile technology. Although there protection in the form on encryption using the 128-bit A5/3 encryption algorithm, which is implemented across all 3G networks.

However, the Isreal’s Wizmann Institute of Science (Lai 2010) has found a way to crack this algorithm[6], thus making it unsafe and susceptible to attacks, likewise to unsecured wireless access points.

[5] Arthur, Charles. *Watchdog finds public Wi-Fi hotspots open to hackers.* October 29, 2009. http://www.guardian.co.uk/technology/2009/oct/29/wifi-watchdog-hotspot-security-vulnerable

[6] Lai, Richard. *3G GSM encryption cracked in less than two hours.* January 15, 2010. http://www.engadget.com/2010/01/15/3g-gsm-encryption-cracked-in-less-than-two-hours/

Thus, due to the vulnerability in this system, there is a need to create a more secured authentication system to prevent theft of passwords.

2.3. Problem of sending over password hashes

The issue of sending over password hashes arises from the fact the information sent over is all that is necessary for a hacker to masquerade as a legitimate user. For example, in a normal authentication, the user sends over his username and hashed password. What happens is that if a hacker is able to sniff the username and hashed password, he will be able to login as the user at any point in time. However, a more pressing issue with sending over password hashes revolves around the insecurity of revealing a plaintext password.

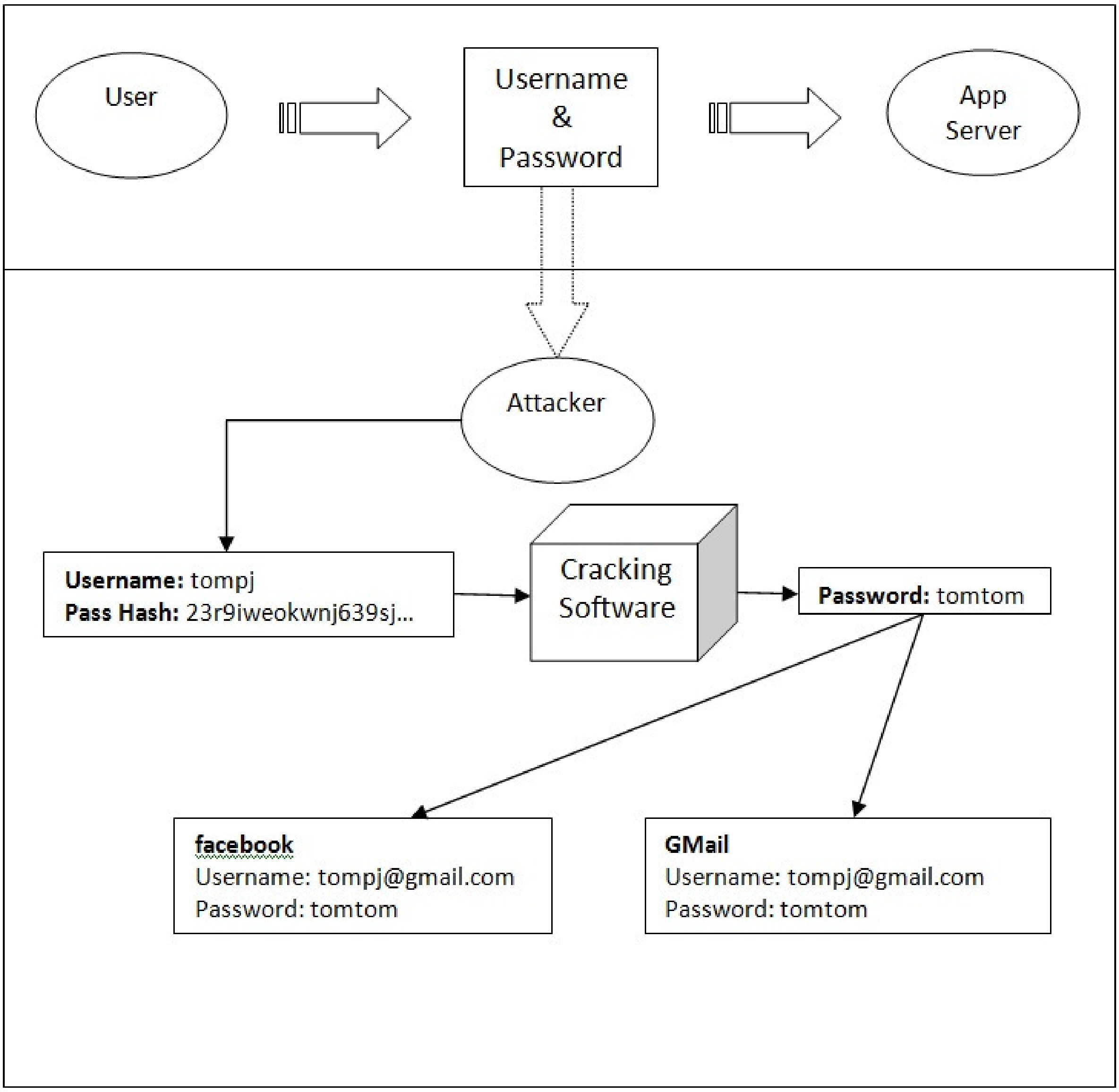

Fig. 2.2. Example of the problem of sending over password hashes

Through sniffing the network, the attacker will be able to obtain the user's password hash. This becomes malicious when the user uses the same password for different services. (I.e. Username: tompj, E-Mail: tompj@gmail.com, etc.)

3. What is ZKA_wzk?

This section will contain information about ZKA_wzk, the logic behind it and how the process changes with the implementation of it.

3.1. Summary of usage of ZKA_wzk

Zero-Knowledge Authentication with zero knowledge (ZKA_wzk) is as created framework which allows companies to easily implement the use of Zero-Knowledge Authentication, which allows for secure login without the need of transmitting the password or hash over the network. This system also allows for the server to prevent storage of password hashes in the database. Thus, if a hacker is able to obtain the database, he/she will still not be able to crack the passwords. The algorithm behind the system is based on the Zero Knowledge Proof of Knowledge (ZKPoK) about Discrete Logarithms – Knowledge of Discrete Logarithms. This is based on the problem of Discrete Logarithms.

Please refer to the thesis "Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem" by Jan Camenisch (Camenisch 1998)[7] for more information on the algorithm used.

3.2. ZKA_wzk Logic Summary

The ZKA_wzk logic, as mentioned will be based on the ZKPoK Sigma Protocol. The following is a step-by-step procedure of the protocol, using $\mathbf{SPK}_1\{(\mathrm{x}) : Y = g_0^x\}$ (a).

Initialization:

1. Given group $\boldsymbol{G}$. Let g_0, g_1 be random elements of $\boldsymbol{G}$.
2. Let the public key be $zkapk = \{\boldsymbol{G}, g_0\}$.

Registration Process:

1. User inputs *username* and *password.*
2. The user hashes the password with Hash function, H and calculates $x = H(password).$
3. The user then computes $Y = g_0^x$
4. The user sends (*username*, Y) to the server

[7] "Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem." Chur: Jan Leohard Camenisch, 1998.

5. The server stores (*username*, *Y*) into the database.

Authentication Process:

1. The server generates a random one-time token *a* and stores it and sends it to the user.
2. User inputs username and password
3. The user hashes the password with Hash function, *H* and calculates $x = H(password)$.
4. The user then computes $Y = g_0^x$
5. The user generates random $r_x \in \boldsymbol{G}$ and calculates $T_1=g_0^{r_x}$.
6. The user then calculates $c=H(Y,T1,a)$.and $z_x = r_x - cx$
7. The user sends (c, z_x) over to the server
8. The server calculates $T_1=Y^c g_0^{z_x}$ and verifies that $c=H(Y,T_1,a)$
9. If successful, user is authenticated.

3.3. How does it work?

The algorithm above is based on a non-interactive sigma protocol. This is a technique commonly used to prove the knowledge of a variable, which in this case, is the password.

In this explanation of ZKA_wzk, let us take the user who is logging in to be the prover and the server verifying the login to be the verifier.

To start off, here are some of the explanation of the components in the algorithm:

Component	Description
G	This is a cyclic group. This group contains a set of numbers which is based on a formula. This is a public group which will be available to both prover (user) and verifier (server).
g_0	A generator of the group **G**. It is an element of the group **G**. This is a public variable which will be available to both prover (user) and verifier (server).
x	The hash of the password that the user inputs
Y	The pseudonym of the user. This is used for the verifier in the calculation of the proof of knowledge.
a	The random token generated for each login attempt
T_1,r_x,z_x,c,	Other miscellaneous variables which are used in the calculation.

Legend:

🟩 Public Values - Known to both the user and the server

🟦 Server Secret Values – Known to the prover, but can be derived from accurate credentials

🟥 User Secret Values – Known only to the user

With that knowledge, here is how the algorithm will work. To keep things simple, you may think of G as a group of natural numbers from {1,2,3,4...}, and g_0 as a randomly generated number of that group, 5. However, in the algorithm, cyclic groups are used, making it difficult to derive the calculation of a discrete logarithm (i.e. finding x where $Y=g_0^x$).

The 3 main processes are:

- Initialization
- Registration
- Authentication

Initialization:

This is where the public key is created. Simply, the value g0 is generated.

For simplicity, let $\mathbf{g_0=5}$.

Registration:

This is where the pseudonym of the user is created. This is simply done by hashing the user's password into x, and calculating $Y=g_0^x$.

For simplicity, let us assume the password is abc and the hashed value is 6.

Thus, **x=6** and $Y= g_0^x =5^6$. Therefore, **Y=15625**. The pseudonym Y is then passed to the server and stored.

Authentication:

Before proceeding, we may now look at what each party has.

Prover (User)	**Verifier (Server)**
$\mathbf{g_0}$	$\mathbf{g_0}$
password	**Y**

In the following, let H be a hash function, i.e. SHA1. For simplicity reasons once again. Please assume that all the following variables are just ordinary natural numbers.

Authentication Process

No	User (Prover)		Verifier (Server)
1			Generate random **a**
2	Receive a	←	Send a
3			
4	Calc. x=H(password)		
5	Calculate $Y=g_0^x$		
6			
7	Randomly generate r_x		
8	Calculate $T_1=g_0^{rx}$		
9	Calculate $c=H(Y,T_1,a)$		
10			
11	Calculate $z_x=r_x - cx$		
12	Send c, z_x	→	Receive c, z_x
13			Calculate $T_1=Y^c g_0^{zx}$
14			Check if c= H(Y,T1,a)

The 'magic' in this formula happens in step no. 14. Notice how T_1 is calculated without the knowledge of the randomly generated secret. r_x?

If we look at the formula of T1 in step 8 and step 13

$$\underline{\text{Step 8:}}\ T_1=g_0^{r_x}$$

$$\underline{\text{Step 14:}}\ T_1=Y^c g_0^{z_x}$$

We will have to prove that

$$g_0^{r_x}= Y^c g_0^{z_x}$$

With reference to step 5 and step 11, we know that

$$\underline{\text{Step 5:}}\ Y=g_0^x$$
$$\underline{\text{Step 11:}}\ z_x=r_x - cx$$

Therefore, by performing a simple substitution, we can prove that

$$g_0^{rx}= Y^c\ g_0^{zx}$$

$$g_0^{rx}= (g_0^x)^c\ g_0^{(r_x\text{-}cx)}$$

$$g_0^{rx}= g_0^{cx}\ g_0^{r_x\text{-}cx}$$

$$g_0^{rx}= g_0^{cx + r_x\text{-}cx}$$

$$g_0^{rx}= g_0^{r_x}\ \textbf{(Proven)}$$

With all 3 elements, we can verify that c=H(Y,T$_1$,a), thus proving that user knows x

3.3. ZKA_wzk Architechture

Registration Process:

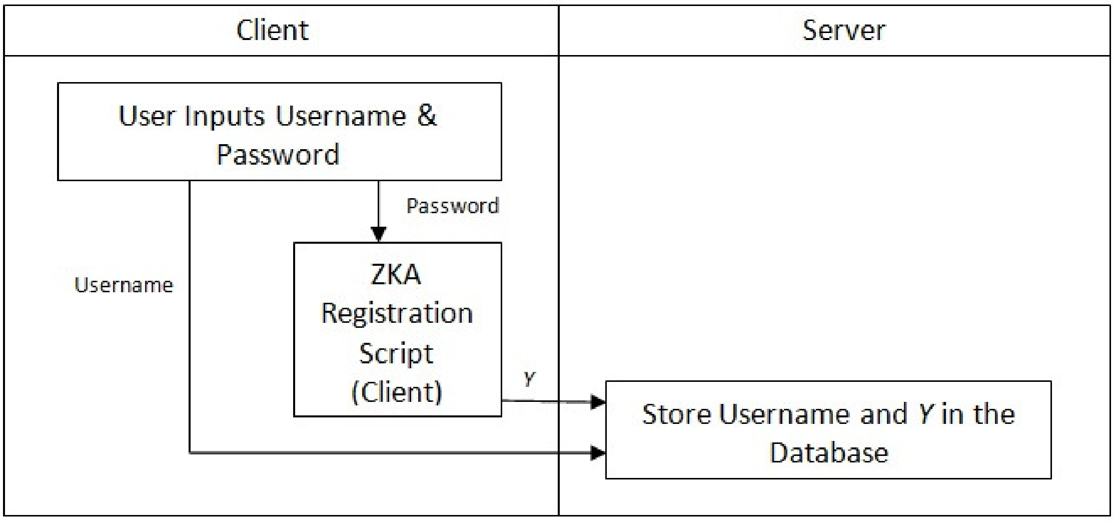
Client
Server
User Inputs Username & Password
Password
Username
ZKA Registration Script (Client)
Y
Store Username and *Y* in the Database

Fig. 3.1. Workflow of the login process

Based on this registration process, we can see that the password hash or password is not stored on the database at all. Even though attackers manage to obtain the username and *Y*, they are not able to compute the password hash, *x*, based on the problem of discrete logarithms.

Authentication Process:

The user firsts requests for the login page, i.e. `login.php`. This page will then pass back a one-time token a, which is stored in the server database (for that session), and passed back to the user. This will be used in the hash later on to prevent using the same, valid credentials.

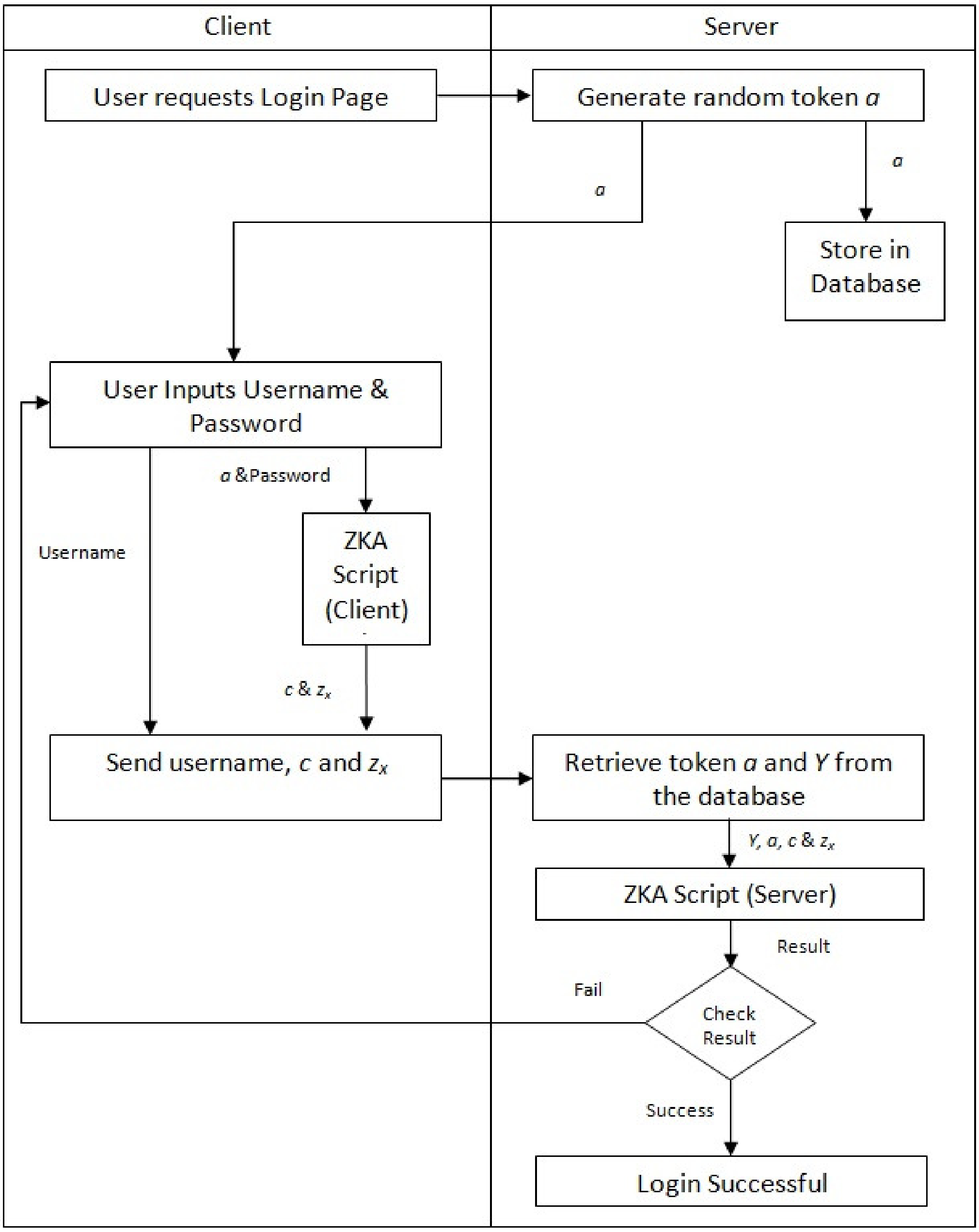

Fig. 3.1. Workflow of the authentication process

From this we can also see that whatever is sent over, c & z_x, has no meaning to the attacker. Even if he understands the formula, it is impossible to reverse a hash and obtain x from z_x. Thus, we can see that the user can easily prove his/her identity without having to send over any confidential data.

4. Why ZKA_wzk?

This section covers benefits of ZKA_wzk, such as the security, ease of implementation and comparisons to other authentication systems.

4.1. Security of ZKA_wzk

ZKA_wzk is secure for several reasons. This includes:

- Data transmitted over the network from authentication is useless to attackers

  The data transmitted over the network, (c, z_x), are not usable by the hacker to help fake an identity

- Prevention of obtaining password hashes or plaintext passwords from sniffing the network

  Obtaining any information being sent over the network, namely, Y, c and z_x, will not allow the attacker to be able to crack the plaintext password of the user.

- Prevention of similar values used through one-time tokens

  Through the use of a one-time token in the hashing function, the information sent over is only valid for once, and thus will not be usable by attackers who intercept the information.

4.2. Comparison of ZKA_wzk with other alternative authentication systems

ZKA_wzk has benefits over other authentication systems due to the fact that there is no additional hardware required, as compared to systems such as biometrics and token-based authentication.

For example, biometric authentication systems rely on an external biometrics reader which not all users may have. Additionally, it has a certain level of uncertainty in its use and often requires a two factor authentication, which still requires using a password.

As for token-based/donger-based authentication, it requires additional hardware and can often result in identity theft from the lost of the identification device if not implemented correctly.

5. The Framework

This section covers how the structure of ZKA_wzk, its components, and how it can be used in web applications.

5.1. The Infrastructure

Similar to other web applications, the platform required for the ZKA_wzk is similar to that of a normal web application. This would consist of a client and a server. An example would be as of below. (i.e. an apache web server and a MySQL database using PHP pages).

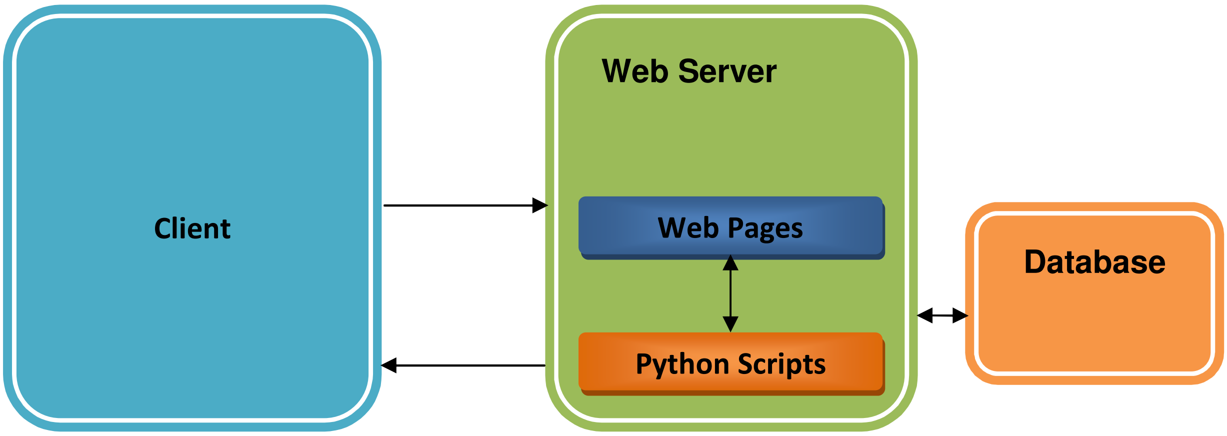

Fig. 5.1. The base infrastructure

Client

As the objective is to prevent the transfer of a password over the network, pre-processing must be done on the client side to calculate the values (c and z_x) and send them over. In order to do this, 3 components need to be present. We shall refer to them as the:

- Interface
- Processor
- Algorithm

In this case, they are a Web Browser, a Java Applet and Python Scripts.

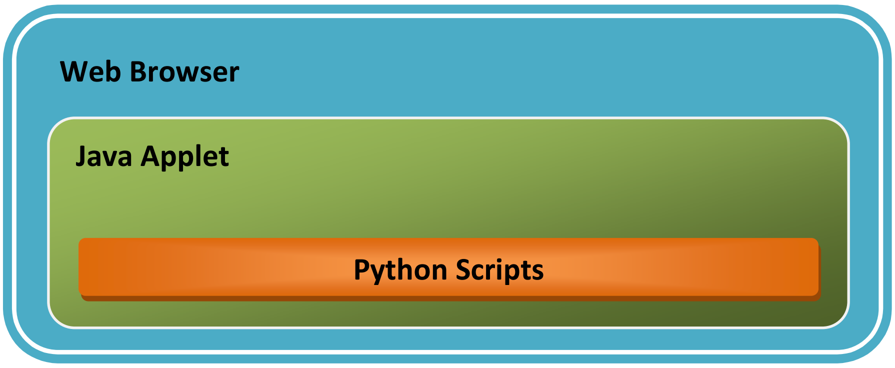

Fig. 5.2. The client overview

5.2. About the code

There are several things to cover in the framework, which are:

- Python scripts
- Java Applet Processor
- Running the process through a web browser

For those who would like to make modifications to the algorithms, changes can be made to the python scripts. However, the way the framework is made is that the client can directly run the python scripts through javascript via the java applet, and post the results to the server, where they are input to the script to be processed. Thus, allowing it to be used without prior knowledge of the formulae.

Python Scripts

There are several python scripts, with each representing an individual function required in the process of the initialization/authentication. This includes:

- initialize.py – initializes generator $\mathbf{g_0}$
- generate_a.py – generates random token **a**
- register.py – calculates value **Y** based on password provided
- userauth.py – calculates **c** and $\mathbf{z_x}$ to be sent over based on user input and token
- vertifyuser.py – calculates and checks the validity of the authentication

Java Applet Processor

The java applet is basically used as a tool to download the python scripts automatically so that it can be used by the web browser via javascript. This is used to make the download and running of the script seamless.

Running the process through a web browser

Through the use of the java applet, they python scripts can be run and controlled easily via javascript. This allows for easy integration and smooth integration onto web applications.

5.3. How to modify the algorithm

To modify the algorithm, the python script files can be edited. The python binding pypbc and its dependencies will be required. The source contains a modified version which allows conversion of the elements to a readable/storable output.

Example: useraut h.py

At the start of the script there will be a variable called stored_params. This defines the group G.

```
   stored_params = """type a
   q
87807107996633125224377819847540498158068831994142082110286
5339926647563088022295...
...
...
   exp2 159
   exp1 107
   sign1 1
   sign0 1
   """
```

The values can be stored and read in later. This is used to read in arguments as well as static values such as g_0.

```
   _g0buf="56:88:1:174:167:6:55:...”
   _g0 = Element.from_bytes(_g0buf,pairing, G1)
   _abuf=sys.argv[1]
   _a = Element.from_bytes(_abuf,pairing, Zr)
```

The algorithm can be modified by editing the arithmetic statements

```
   #Step 4 - calculate Y
   _Y=pow(_g0,_x)
   #Step 5
   #Generate rx
   _rx = Element.random(pairing,Zr)
   #Calculate T1
   _T1=pow(_g0,_rx)
```

The output can then be printed out. This is to allow the variables to be passed from client to server and vice versa, as well as to store the static variables.

```
   print(Element.to_bytes(_c))
   print(Element.to_bytes(_zx))
```

5.4. How to use it

In the authentication process, scripts will have to be processed on both the client and server side. Below shows how these scripts can be run in very brief details. The source files will provide a working example to start with.

Server-Side

On the server side, the python scripts can be run via a web application. For example, using PHP with the variables from authentication:

Register

```
  $input = "./register.py "."'".$pass."'";
  exec($input, $result,$rtv);


  $_Y=$result[0];
  include("config/mysql.php");
  $sql = "SELECT username FROM `ZKAwzk`.`Users` WHERE
       Username = '$user'";
  $result = mysql_query($sql, $link);
  mysql_close($link);
```

Generate token (to be placed on the login page/frame)

```
exec("./generate_a.py", $result,$rtv);
echo "<BR><BR>Value of token (a):<BR>$_a";
```

Authenticate

```
$input  =  "./verifyuser.py  "."'".$_Y."'"."  "."'".$_a."'"."
"."'".$_c."'"." "."'".$_zx."'"." ";
  exec($input, $result,$rtv);_
  print $result[0];
```

Client-Side

The code can be easily used through having a login form as usual, but adding in several other variables.

```
<form name=login method="post" action="authenticate.php">
<input type="hidden" name="_a" value="<?php echo $_a?>">
Username:<input type="text" size="12" maxlength="36"
name="user">
Password: <input type="password" size="12" maxlength="36"
name="pass">
<input type="hidden" name="_c">
<input type="hidden" name="_zx">
</form>
```

The java applet also needs to be loaded in the page:

```
<applet name="ZKAapplet"
    code = "ZKA.run.class"
    archive = "applet.jar"
    width = 120
    height = 25 >
    <param name="token" value="<? echo $_a?>"/>
    </applet>
```

Next, to run the script with the form values, a javascript call can simply be done:

```
<script type="text/javascript">
  document.ZKAapplet.authenticate(document.login.user.value
,document.login.pass.value);
</script>
```

Thereafter, the values can be set and posted via the form:

```
<script type="text/javascript">
  document.login._c.value = document.ZKAapplet._c;
  document.login._zx.value = document.ZKAapplet._zx;
  document.login.pass.value="";
  document.login.submit();
</script>
```

6. Conclusion

Besides providing a much higher level of security to a web application, there are many other reasons why ZKA_wzk is worth implementing. Firstly, it allows for someone with no knowledge on how the protocol works to take advantage of such a concept. Also, through the use of embedded applets, the system is basically transparent to the user. Finally, the simplicity and ease to implement the system is definitely a value-add and a reason to have this in any web application.

7. About the author

Brandon Lum Jia Jun is a graduate of the diploma in Cyber and Digital Security/Temasek Informatics & IT School. Despite only becoming part of the IT industry in 2007, he is an active member of the security community in Singapore. As a security enthusiast and firm believer in Open Source technologies, he has helped organize security and open source seminars and events over the past three years. He is also the recipient of the Lee Kuan Yew Maths and Science Award.

Brandon has had experience in implementing cryptography systems during a 3-month term in which he was attached to the University of Wollongong for research under Professor Willy Susilo. There, he implemented systems based on cryptography techniques such as Zero Knowledge Proof of Knowledge and BBS+ Signatures.

In addition Brandon is part of the team that emerged champion and top offensive team at a security based competition, Syscan ’09 CTF (Capture The Flag). He is also the team lead in a project which has achieved the SiTF and APICTA Award and the team leader of the champion team of a Secure Coding Competition, HelloSecure@SG. He has also obtained the Singapore Top IT Developers Award, which is given to the top developers in Singapore.

8. Acknowledgement

I would like to give special thanks to my supervisors, Professor Willy Susilo, A/Professor Yi Mu, and Dr. Allen Au for giving me the chance to learn and understand the Cryptography concepts during my time in the University of Wollongong (UoW). I would also like to thank Temasek Polytechnic Informatics & IT School for providing me the opportunity for the attachment to UoW.

9. References

Arthur, Charles. *Watchdog finds public Wi-Fi hotspots open to hackers.* October 29, 2009. http://www.guardian.co.uk/technology/2009/oct/29/wifi-watchdog-hotspot-security-vulnerable (accessed December 27, 2009).

Grossman, Jeremiah. *Be Ready -- With Answers.* February 2, 2010. http://jeremiahgrossman.blogspot.com/2010/02/be-ready-with-answers.html (accessed February 7, 2010).

"Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem." Chur: Jan Leohard Camenisch, 1998.

"How to Explain Zero-Knowledge Protocols to Your Children." *Advances in Cyptology - CRYPTO '89.* Jean-Jacques Quisquater, Louis C. Guilou, Thomas A. Berson, 1990. 628-631.

Lai, Richard. *3G GSM encryption cracked in less than two hours.* January 15, 2010. http://www.engadget.com/2010/01/15/3g-gsm-encryption-cracked-in-less-than-two-hours/ (accessed February 20, 2010).

Vijayan, Jaikumar. *Web application security is growing problem for enterprises.* November 11, 2009. http://www.infoworld.com/d/security-central/web-application-security-growing-problem-enterprises-843 (accessed December 31, 2009).

Yadav, Sujeet. *Rising Popularity of Web Application Development* . August 28, 2008. http://ezinearticles.com/?Rising-Popularity-of-Web-Application-Development&id=1449765 (accessed December 28, 2009).