Secure Login without Passwords

T.M.C. Ruiter, $X \oplus R_{\text{key}}$

February 3, 2014

Abstract

These are the ingredients for secure logins:

- For login purposes, the webserver uses a separate login server. This server can be internal (in the back-office) or external (somewhere on the Internet).
- The login server maintains a small array with Secret keys in an HSM, which will be renewed regularly (oldest key removed, all keys shifted one position, new key added).
- All cryptographic operations for the login procedure are performed by the login server and take place inside the HSM; no values are remembered afterwards, except Secret keys.
- For each user, a random number acts as the site key for that user. The user gets a different value, which is the $\texttt{AES}_{256}$ encryption of the site key for that user with the newest Secret key.
- Logging in is a process of proving that both the website and the user have the right key by sending $\texttt{SHA}_{256}$ hashes of random numbers encrypted with these keys. Neither the site key nor the user key are ever sent over the line.
- When Secret keys are changed, the user automatically gets a new key. This way, user keys are changed regularly as well.
- All user keys are put on a keyring, which is a set of at least 99 keys, most of them dummies. No other information whatsoever is stored in a keyring, which may be stored in an encrypted file.
- The user selects which key is used for what, which it needs to write down or remember. Although keys themselves are changed regularly, the key number and its purpose will never change during the lifetime of the keyring, which might be forever. This makes remembering key numbers, at least for those keys that are used regularly, doable for most people.
- The keyring itself is something you must have; the key number something you must know, so logging in using a keyring is a basic form of two-factor authentication.

1 Introduction

Passwords should be kept in memory—human memory, that is—at all times. This article is about passwords for websites; the passwords many people use on a day to day basis. Passwords are an archaic type of security measure ([2]), compared to the scheme proposed here.

1.1 On human behavior

The rationale amongst security experts is that passwords should not be written down. Ever. And passwords should be unique for each and every website. Oh, and—I almost forgot—you have to change them as well. Each month or so will do nicely.

Yeah, right! I cannot remember all different passwords I am forced to use, although my memory is quite good. I *have* to write them down, otherwise, I am lost. (Fortunately, I have some support for this [3].) Therefore, I resort to a little black booklet, with all my account information. I don't remember passwords any more, I remember where my booklet is. And I confess that I am compelled to reuse passwords, and to keep the ones I am not forced to change, so I can actually remember some of them. So I believe nothing has fundamentally changed in more than 14 years... [1]

I have given up on inventing a scheme for passwords that differ from each other, can be changed individually at different times, and are easy to remember, as not to be forced to write them down. I believe no such scheme exists, because websites have different requirements regarding passwords. To see what I mean, watch [4].

For the user, the bad thing with passwords is that you have to keep track of it all. Remembering difficult passwords is cumbersome for most, and impossible for some. Tracking things infalliably, and remembering different passwords for each and every site is not something people excel at.

1.2 A new login approach

Using a key, with up to 128 random bits, to gain access to a website is far more secure than letting people decide which password they would like to use to do so.

This proposed new way of logging in has several advantages over the current practice of websites requiring passwords. Instead of having to remember dozens of passwords for numerous sites, you only need to remember a key number for that site, in the range of 1 to 99. This key number stays the same for that website at all times, so you *can* remember it.

2 Login for dummies

Not that you are one if you decide to read this section. It serves as a general description of the login process, without going into technical details too much.

2.1 A new login scheme

Userids tend to be in short supply for people with first names like John, Peter, and Chris, or surnames like Jones or Smith. This is true for people in almost any country. And the larger the site, the rarer free userids.

Passwords tend to be weak, as people choose short, real-life words, like things or names. Not that people are not willing, but remembering too long, too difficult passwords is not easy. It is very frustrating when you cannot login because you have forgotten your password, or, just when you are starting to remember it, you need to change it again.

As a radical measure, we do away with all this!

Instead of userids, the website holds hashes based on keyring identifiers. Instead of a password for an account, the website holds one part of a key, of which you have the other part, like the broken locket Annie clings to in the musical that bears her name. Your key is different from the key the website has, but they are related to each other. It is not possible to login with the key that is stored on the website; you can only login with your personal key, which is verified with the key from the website.

We will let the computers (yours, the one running the website, and another one dedicated for logins) do what they do best: compute. Give them some random bits to chew on, and they are in heaven. We will let humans (you, your neighbor and your fellow earthlings) do what they do best: remembering small ordinal numbers.

2.2 Storing your keys

Keys are just long strings of random bits with no information whatsoever. Your keys are personal and are stored locally.

Imagine a keyring, not unlike your own keyring on which you have keys for your car, your house, shed, or locker. This keyring has a label and 99 keys on it, numbered 1 through 99. Instead of brass or steel, these keys are made of 128 random bits each. The label is another 128 bits long, and as random as possible, like the keys. Instead of being on a steel ring, these keys, with the keyring label, are written to a file on disk; a blob of 100 strings of 128 bits.

The use of keys on this keyring is not entirely different from using real physical keys. The bits in a key are comparable with the tooths or holes of a physical key you use to unlock your home. You do not need to remember exactly how far the tooths need to protrude or exactly where and how deep the holes in your key need to be, to be able to unlock the door; you just select the right key (the whole physical thing at once, with all the right tooths or holes) by recognising its form or its label.

2.3 Computers and links

This proposed way of logging in concentrates on computing values and communication of the results. It is therefore appropriate at this point to elaborate a bit on who is communicating with whom and why.

2.3.1 Computer roles

Four computer roles can be distinguished when logging in.

Webserver This is the front-end server to which the user is communicating. From our point of view, it is the central system, acting as a hub. It communicates with three other servers: the accounts server, the login server, and of course the client computer. We consider it the most vulnerable to 'visits' with malicious intent, so it has been appointed the most simplests of tasks. It does not compute anything remotely valuable, it just compares values provided by others.

Accounts server This server is typically located in a network only accessible by the webserver. It stores all kind of user related account data, but no userid or passwords; it stores hashes and site keys instead. Like the webserver it does not compute anything either.

Login server This server uses an HSM which stores an array of secret keys per website it services. This server computes nothing on its own, it lets the HSM do all the work secretly, using some of its finest cryptographic functions. It receives login requests from the webserver and returns values that the webserver can compare or relay.

Client computer With this device the user communicates with the webserver. It generates random numbers and uses `XOR` and $\texttt{SHA}_{256}$ to compute the values exchanged with the webserver.

These roles can be played by as many computers as there are roles, but that need not always be the case. In fact, each and every role may be combined; a single computer may perform them all, for that matter.

The accounts server and the login server may be combined in a single server in the back-office. The webserver and the accounts server may be combined, with the login server somewhere on the Internet. The combination of webserver, accounts server, and login server is also valid; although this concentration of roles is not highly preferred.

2.3.2 Secure links

Traditionally, the communication channel between the user and the webserver is secured by means of TLS. The user sees the URL that starts with "https://", which means that the webserver has authenticated itself using a digital certificate. The browser will validate the supplied certificate using other trusted certificates. This security measure is sufficient but required.

The communication between the webserver and the accounts server should be conducted over a secure channel as well.

Finally, the communication between the webserver and the login server needs to be encrypted. For this, TLS is a sufficient security measure, although in this case mutual authentication is a must. The webserver needs to know it is talking to a legitimate and trusted login server, and the login server needs to know it is receiving valid requests. The certificate of the websever is used to select which array of secret keys to use, as the login server may service more than one webserver.

2.4 Logging in

2.4.1 Part I

Before starting the login process you select a key from your keyring; the one you know to belong to the website you are trying to login to.

To login you don't send your key to the server but generate a secret random value, which you encrypt with your key using a simple `XOR` operation. Your key is totally random and the secret random value you encrypt with it is also, well... totally random. Mixing these random bits gives another totally random value. The website will receive this value and will try to decrypt this with the key that belongs to your account (we will keep you in the dark for now, about how this works). When it has decrypted the random login value, it will know which secret random value you generated. Instead of telling you the secret random number, the website sends a hash value of it, because otherwise someone able to see the network traffic will instantly know your key.

If the website returns a hash value matching your secret random number, you know two things. First, you know that the website you are talking to can successfully decrypt your random value. It can only do this if it has a matching key. Second, you know you are talking to the same site that sent you your key when you applied for an account. This implies that if you trusted that site then, you can trust this site now, as it can only be the same site.

2.4.2 Part II

This is all great and very sophisticated, but the website wouldn't dare share your wonder about this, as you may well fake your enthusiasm, and flatly lie about the correctness of the hash it has sent you. To see if your claims hold, the website will do the same as you and send you an encrypted secret random value. This cryptogram can only be decrypted by someone owning the right key for the account. Using `XOR` with the key you have originally selected from your keyring, you decrypt it. Then, using `XOR` again, you combine the two random values you now have, and return this value to the website.

When the website receives a value matching its secret random number, it knows two things. First, the user on the other side can successfully decrypt the secret random number. It can only do this if it has a matching key. Second, the website knows that this key is from the same keyring that was used when applying for an account. This implies that if you trusted this user then, you can trust this user now, as it can only be the same user.

2.5 Keys

2.5.1 On site keys and user keys

Beware, the icky parts start here (a bit).

When applying for an account, a site key is created by a random generator. The user key is then computed as the encryption of the site key with a Secret Key, with a block cypher like $\texttt{AES}_{256}$.

When the user encrypts its secret random value with its key, it uses `XOR`. This is a very simple and reversable encryption method. But as both key and value are utterly random, no information is stored in the encrypted value. The website needs

to decrypt the value, and this can only be done with the user key. But the website does not store user keys, only site keys...

The solution to this may be a bit of a dissappointment and a cheat: the user key is temporarily regenerated by encrypting it again with the Secret Key. Once the user key is available, the random value can easily be decrypted by `XOR`-ing it. Also, the cryptogram that is sent to the user is a random value `XOR`-red with this regenereated user key.

The clever part, however, is that all the crpytographic functions the website is required to perform take place inside a Hardware Security Module, or HSM for short. Secret Keys can be put in the HSM and made to good use there, but can never be retrieved from it. The HSM computes all values needed for the login process, without ever revealing the keys that are used, not even to a hacker with full control of the HSM.

In the end, only random bits enter the HSM, and only random bits leave it.

2.5.2 Key lifetime

Every key has its lifetime, so Secret Keys need changing every so often, as a good security measure. The same goes for keys on the user's keyring.

The login protocol is capable of changing keys on the website and keys on the keyring, without any effort on the user side. Well..., a little program will do it for you, without you noticing.

Logins can be granted, but new keys may not be, which results in the expiration of an account. A website may deny a user new keys when payment is due, giving users limited login rights. At any time new keys can be provided again.

To learn how this all works, how keys are used, what a website can do with logins, and more, can all be read in the next chapters.

3 Conclusions

The security of the login process for websites can be greatly improved by using a keyring at the user side and an HSM employed by the website. Instead of sending relatively short, easy to guess strings (passwords) over the line, the use of encrypted random values, up to 128 bits each, is a big improvement. No keys are sent, just random values, which will be different each time a user logs in.

For hackers, getting login data in huge numbers will be very difficult, since this data is no longer stored centrally, but split between website and user. Each part alone has no value, and all values stored at the website render no valid login data without Secret keys. These keys are kept in very secure hardware: an HSM. Sniffing network traffic or collecting keystrokes with Trojans will not help. Keyrings have very high entropy and are encrypted, so to no direct use to hackers as well; they may be stored on the web for easy access and backup.

Several important security measures are automatically implemented: keys are changed frequently (as frequent as Secret keys are changed), they differ for each website, and keys on a keyring and the knowledge which key is used for what site constitutes two-factor authentication.

For the user, the way to logon to websites will change, but it will be an improvement over the burden of keeping track of all passwords. One userid for all sites and a single key number for a website is all you have to remember.

3.1 Advantages

In the following cases a keyring is superior to the use of passwords.

- Websites have all login information stored centrally. If an hacker can obtain this data and decrypt it, it has access to all—possibly millions—accounts at once [5].

  Using the keyring system there is no usable login information whatsoever at the server hosting the website. There is no way that the user information that is *present yield any usable login data. Hacking a website to obtain logins is useless. Other reasons to hack websites will remain, however, and using keyrings does not prevent hacking; it just eliminates one of the major attractions.*

- Users tend to have the same password and the same login name for several websites. A hack of an insufficiently protected site could yield valid usernames and passwords of perfectly protected sites. (Hack of www.babydump.nl yields at least 500 valid logins for www.kpn.nl.)

  Even if all user keys for a website were obtained in a hack, these would be useless for any other website, since they differ by definition.

- Websites require the user to change passwords. As more websites do this, more and more passwords a user has to remember, change. Ideally, no password for a website should be the same as for another website, but that is impractical. This would mean that each and every password needs to be written down, because the number of passwords is too much to remember for most. This thwarts the principle that passwords need to be remembered and never written down. The requirements to change passwords frequently and that they should differ from any other password is an inhuman task.

Using the keyring system, keys will differ for each website by definition and change regularly and automatically. Key numbers (the ordinal numbers in the keyring) don't change, so most of them can be remembered by the average human.

- Sometimes, getting unauthorized access to an account is as simple as just looking at the keyboard to see what the password is. The userid is always displayed when logging in, so shoulder surfing is very effective.

  Using a keyring, shoulder surfing cannot be used directly to login. Since a keyring is something you have to have, you cannot login using only the userid and the key number. You need to have access to the (unencrypted) keyring as well. Therefore, using a keyring is a basic form of 2-factor authentication.

- People tend to use weak passwords (unless a website specifically enforces the use of strong passwords) which can be guessed using specialized tools. If that yields no success, brute force attacks can be launched; to just try all possible passwords with limited length.

  Password guessing nor brute force attacks are an option when trying to login, since no passwords of any kind are exchanged. Even if the keys themselves would be used as an old-fashioned password, the search area would encompass 2^{128} *or* $3.4 \cdot 10^{38}$ *equally likely possibilities. Trying 1 million possibilities per second it would still take* 10^{32} *seconds to try them all.*

- The validity of the connection to websites is built on trust. HTTPS connections are protected using certificates. Sometimes trust only goes so far, and bogus but valid website certificates are used (Dorifel virus) or even the Root CA certificates are forged (see the DigiNotar hack). In that case, the user's trust is betrayed and the user left helpless.

  With the keyring solution no standalone substitute websites can exist; login data must be redirected to the real website. A substitute website does not have the right Secret keys. A user will notice this by wrong answers from the website and the login will abort from the user side.

- Visitors of websites are lured to other, well built fraudulent websites, mimicking websites of banks and such (phishing). Here, a simple mail can give a lot of trouble, redirecting users to an unsecure copy of a website, without the user suspecting anything.

  All communication to and from the malicious website can be passed on to the real website, to give the user the sense it is talking to the real site. Obtaining valid login data this way (as a man in the middle) is useless, since no keys are sent over the line. The data that is used to validate a user is meaningless for the next login.

- People are sometimes called by other people, claiming to be employees of banks. In order to "help" solve a problem, users are asked to give their login credentials. Some ignorent users are willing to oblige.

  With a keyring the only thing slightly useful to a hacker would be the userid. The keynumber is of no use, since the hacker has no access to the keyring itself; telling him which key index is used to login has no value.

Acknowledgements

Thanks to Martijn Donders for his cryptographic support, and Rob Bloemer for reviewing the first draft of this article. The review sessions with Jannes Smitskamp were pleaseant and intense, and helped a lot to expose some hidden flaws; which were all remedied elegantly.

References

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, December 1999.

[2] Mat Honan. Kill the password: Why a string of characters can't protect us anymore. *Wired Magazine*, 11 2012.

[3] Bruce Schneier. Write Down Your Password. Cryptogram Newsletter, June 2005.

[4] Toby Turner. Password rant, December 2012. http://www.youtube.com/watch?v=jQ7DBG3ISRY.

[5] Wikipedia. 2012 linkedin hack — wikipedia, the free encyclopedia, 2012. [Online; accessed 17-September-2012].