How To Sync with Alice

Feng Hao[1] and Peter Y.A. Ryan*[2]

[1] School of Computing Science
Newcastle University
feng.hao@ncl.ac.uk
[2] Faculty of Science
University of Luxembourg
peter.ryan@uni.lu

Abstract. This paper explains the sync problem and compares solutions in Firefox 4 and Chrome 10. The sync problem studies how to securely synchronize data across different computers. Google has added a built-in sync function in Chrome 10, which uses a user-defined password to encrypt bookmarks, history, cached passwords etc. However, due to the low-entropy of passwords, the encryption is inherently weak – anyone with access to the ciphertext can easily uncover the key (and hence disclose the plaintext). Mozilla used to have a very similar sync solution in Firefox 3.5, but since Firefox 4 it has made a complete change of how sync works in the browser. The new solution is based on a security protocol called J-PAKE, which is a balanced Password Authenticated Key Exchange (PAKE) protocol. To our best knowledge, this is the first large-scale deployment of the PAKE technology. Since PAKE does not require a PKI, it has compelling advantages than PKI-based schemes such as SSL/TLS in many applications. However, in the past decade, deploying PAKE has been greatly hampered by the patent and other issues. With the rise of patent-free solutions such as J-PAKE and also that the EKE patent will soon expire in October, 2011, we believe the PAKE technology will be more widely adopted in the near future.

1 Introduction

The past two decades have seen the gradual evolution of a computer. A computer used to be a luxury, but now it is a necessity; it used to be bulky and fixed at one location, but with the rise of smartphones and tablets, it is becoming smaller and more mobile; it used to store data locally, but now data storage is moving to the cloud (which can be accessed anywhere from the Internet).

One trend from this evolution is that an individual now tends to own several computing devices. At home, he may use a good-performance desktop PC for entertainment; on the road, he may use a smart phone to read news and check emails; at meetings, he may use a laptop or a tablet to deliver a presentation. The possession of multiple computers naturally raises a practical problem: how to keep data in sync across different platforms?

* Sponsored in part by the FNR Luxembourg

Dropbox offers a popular solution. According to the report, it has a population of 25 million users worldwide [3]. To set up a sync account, the user needs to provide a username/password. Once installed, the software will centrally store the user's files on the company's servers, automatically tracks the changes, and synchronizes the changes across the user's computers. The sync process happens in the background and is transparent to users.

However, there is a serious lack of privacy protection in the Dropbox solution. As Dropbox states its security policy on its website [3], "Dropbox employees are prohibited from viewing the content of files you store in your Dropbox account, and are only permitted to view file metadata (e.g., file names and locations)." Meanwhile, the company also acknowledges: there are a small number of employees who must be able to access the files whenever necessary. Although this is stated by the company policy as "rare exception, not the rule", the security is hardly reassuring. (If an insider attacker leaks users' personal files to the government, the users will probably never know.)

Browser vendors face exactly the same problem. Every browser keeps a user profile, which includes history, bookmarks, cached passwords and so on. The user profile used to be stored locally, but it has become increasingly necessary to store it remotely (in a "cloud "), and synchronize the profile across the user's computers. This can significantly improve the usability and productivity. For example, if a user buys a brand new laptop, after sync he will be able to instantly re-use the same bookmarks, history etc that were previously accumulated on another laptop. This is quite convenient.

As browser vendors recognize, security is a key issue. The user profile contains security-sensitive information – for example, it may contain passwords for on-line banking or other accounts. If the data is stored on the vendor's "cloud" and the vendor can read data, users must completely trust the vendor (just as in Dropbox) not to misuse it. But, the problem goes deeper than the mere trust. If the vendor has ready access to all the user's on-line account passwords in the cloud, what are the legal implications if the user accounts are compromised? How can the vendor establishes the public confidence that it did not leak any user's passwords nor misuse them?

The right solution seems to have an end-to-end encryption between the two sync computers. All data between the computers is encrypted. The user is the sole holder of his own encryption key; no one else is able to read data – not even the cloud provider. Both Mozilla Firefox and Google Chrome aim to provide such a solution. In the following sections, we explain their solutions in detail. The same sync design in the browser is instrumental and can be generally applied to many other applications (e.g., to address the security loophole in Dropbox).

2 Background

In this section, we briefly explain the Password Authenticated Key Exchange (PAKE) technology in general and the J-PAKE protocol in particular. They are relevant to solving the sync problem.

2.1 Password Authenticated Key Exchange

Password Authenticated Key Exchange (PAKE) is a foundational building block for a wide range of security applications. This technique allows establishing secure communication between two parties solely based on a shared password without requiring a Public Key Infrastructure (PKI). A PAKE protocol shall fulfill the following security requirements:

1. **Off-line dictionary attack resistance** – It does not leak any information that allows a passive/active attacker to perform off-line exhaustive search of the password.
2. **Forward secrecy** – It produces session keys that remain secure even when the password is later disclosed.
3. **Known-session security** – It prevents a disclosed session from affecting the security of other established session keys.
4. **On-line dictionary attack resistance** – It limits an active attacker to test only one password per protocol execution.

A secure PAKE protocol has several compelling advantages over PKI-based schemes such as SSL/TLS. First, it does not require a PKI, which is particularly expensive to set up and to maintain. Second, it allows zero-knowledge verification of a password: in other words, the user can prove to the other party the knowledge of a shared password without revealing it. Since the password is never disclosed to the other party (unlike in HTTPS), a PAKE protocol is naturally resistant to phishing attacks.

The first PAKE protocol was called the Encrypted Key Exchange (EKE), designed by Bellovin and Merrit in 1992 [5]. Subsequently in 1996, Jablon proposed another solution called Simple Password Exponential Key Exchange (SPEKE) [7]. Many other PAKE protocols were proposed. In 2000, IEEE P1363.2 formed a working group to study all available PAKE protocols and to select secure ones for standardization. However, in 2008, the project ran out of the maximum eight years; no concrete conclusion seemed to be made.

Two hurdles emerged during the standardization process. First, patent was a big issue. Many PAKE protocols were patented. In particular, EKE was patented by Lucent Technologies [6], SPEKE by Phoenix Technologies [8], and SRP by Stanford University [4]. Second, these protocols were found vulnerable. EKE was reported to leak partial information about the password, hence failing to satisfy the first requirement [9]. SPEKE was found to allow an active attacker to test multiple passwords in one protocol execution, therefore it does not fulfill the fourth requirement [11]. Similarly, the SRP does not satisfy the fourth requirement, as explained in [12]. None of these protocols have security proofs.

2.2 J-PAKE

It became clear in 2008 that the PAKE problem was still unsolved. In the same year, Hao and Ryan proposed a new PAKE protocol, called Password Authenticated Key Exchange by Juggling (J-PAKE) [1,2]. The protocol follows a completely different approach from past schemes. It works as follows. Let G denote a subgroup of Z_p^* with prime order q, and g be a generator in G. Let s be a shared password between Alice and Bob, and $s \neq 0$ for any non-empty password. The value of s is assumed to be within $[1, q-1]$. Alice selects two secrets at random: $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$. Similarly, Bob selects $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$.

Round 1 *Alice sends out g^{x_1}, g^{x_2} and knowledge proofs for x_1 and x_2. Similarly, Bob sends out g^{x_3}, g^{x_4} and knowledge proofs for x_3 and x_4.*

The above communication can be completed in one round as neither party depends on the other. When this round finishes, Alice and Bob verify the received knowledge proofs, and also check $g^{x_2}, g^{x_4} \neq 1$.

Round 2 *Alice sends out $\mathcal{A} = g^{(x_1+x_3+x_4)\cdot x_2 \cdot s}$ and a knowledge proof for $x_2 \cdot s$. Similarly, Bob sends out $\mathcal{B} = g^{(x_1+x_2+x_3)\cdot x_4 \cdot s}$ and a knowledge proof for $x_4 \cdot s$.*

When this round finishes, Alice computes $K = (\mathcal{B}/g^{x_2 \cdot x_4 \cdot s})^{x_2} = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$, and Bob computes $K = (\mathcal{A}/g^{x_2 \cdot x_4 \cdot s})^{x_4} = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$. With the same keying material K, a session key can be derived $\kappa = H(K)$, where H is a hash function. Alice and Bob will subsequently perform explicit key confirmation as described in [1]. In the protocol, the knowledge proof can be realized by using, for example, Schnorr signature. Overall, the J-PAKE protocol has been proved to fulfill all the four security requirements. In addition, the protocol is unpatented. The J-PAKE protocol and security proofs have been available on the IEEE P1363.2 website[3] for public review for over three years; no attacks have been found.

3 Sync solutions in Browsers

In this section, we will explain how major browser vendors try to tackle the sync problem. In particular, Firefox 4 presents an interesting case study as it is the first browser to adopt the PAKE technology in the sync design.

3.1 Overview

Sync has become an important feature for a modern browser. With the exception of IE 9, new releases of browsers generally have built-in support for sync (see Table 1). In the following sections, we will focus on comparing sync in Firefox 4 and Chrome 10, as their solutions are representative.

3.2 Chrome sync

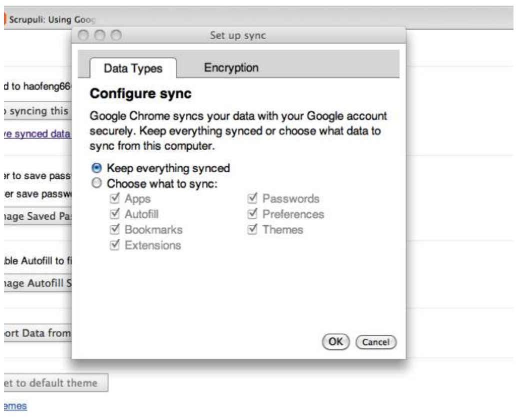
Chrome 10 provides a straightforward sync design, based on using a password as the encryption key. Setting up sync in Chrome 10 is almost zero effort – as long as you have an Gmail account. The user can then configure what to sync. By

[3] http://grouper.ieee.org/groups/1363/Research/contributions/hao-ryan-2008.pdf

Browser	Release date	Built-in	Sync-key	Price
Firefox 4	Mar, 2011	Yes	128-bit	Free
Chrome 10	Mar, 2011	Yes	Password	Free
IE 9	Mar, 2011	No	–	–
Opera 11	Dec, 2010	Yes	None	Free
Safari 5	Jun, 2010	Yes	None	$99 per year

Table 1. Overview of Sync solutions in browsers

default, that is everything: apps, auto-fill, bookmarks, extensions, preferences, themes and passwords (Figure 1). The browser offers two options to encrypt the sync data: re-using the Gmail password (default) or choosing a new password (Figure 2).



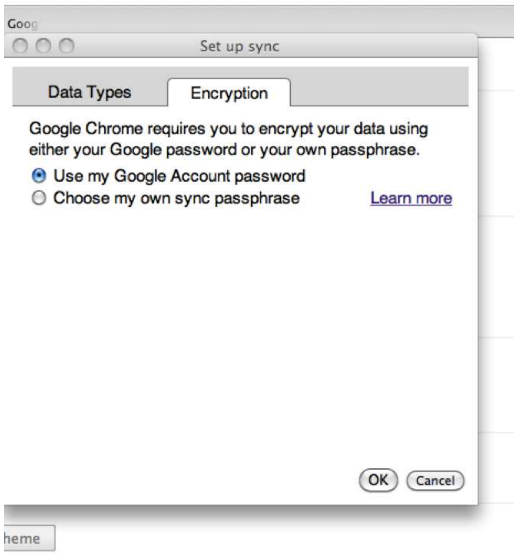
Configure sync

Google Chrome syncs your data with your Google account securely. Keep everything synced or choose what data to sync from this computer.

- ● Keep everything synced
- ○ Choose what to sync:
  - ✓ Apps
  - ✓ Autofill
  - ✓ Bookmarks
  - ✓ Extensions
  - ✓ Passwords
  - ✓ Preferences
  - ✓ Themes

Fig. 1. Confgure sync in Chrome 10

However, Google's solution provides virtually no guarantee of privacy. In both options, the encryption key is directly derived from a password. Due to the human's inability to remember cryptographically strong secrets, a password normally only has 20-30 bits entropy. Thus, although Google encrypts the sync data in its cloud, the encryption key is inherently weak. Anyone who has access to the ciphertext can readily break the key by exhaustive search and fully uncover the sync data.


Set up sync

Data Types | Encryption

Google Chrome requires you to encrypt your data using either your Google password or your own passphrase.

[✓ Use my Google Account password]

Choose my own sync passphrase

Fig. 2. Encryption options in Chrome 10 sync

3.3 Firefox sync

The previous version of Firefox (3.5) used to have a similar sync solution. To set up sync, the user needed to remember two passwords: one for the sync account, and the other for encrypting data. The encryption works basically the same as in Chrome 10 – using a user-defined password as the encryption key. One subtle difference is that in Chrome 10, the default option is to re-use the Gmail password as the key, while in Firefox 3.5, the default is to let the user define a new password.

Because the encryption was inherently weak, Firefox 3.5 had the same problem as in Chrome 10. Similar to Google, Mozilla was at a privileged position: it was able to read all the user's data despite that the data was encrypted (by a password). In recognition of this problem, the company has been trying to find a solution.

From Firefox 4 beta 8 (released in Dec, 2010), Mozilla made a complete change in the sync mechanism. The new solution adopts the Password Authenticated Key Exchange technology – in particular, it chose J-PAKE. Figure 3 shows an overall diagram about how sync works in Firefox 4. First, the browser generates a random 128-bit key, called the sync-key. This sync-key is never sent to Mozilla. It is used to encrypt the browser bookmarks, history, cached passwords etc. Only the encrypted data is stored at the Mozilla "cloud". Alternative servers can be used, and one can even set up his own server.

To set up sync in Firefox 4 is relatively straightforward. First, one needs to configure what data to sync (see Figure 4). Second, the J-PAKE algorithm is used

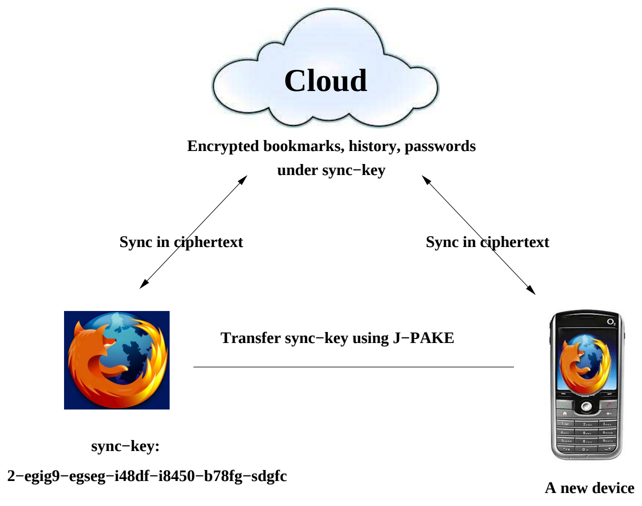

Cloud
Encrypted bookmarks, history, passwords under sync−key
Sync in ciphertext
Sync in ciphertext
Transfer sync−key using J−PAKE
sync−key:
2−egig9−egseg−i48df−i8450−b78fg−sdgfc
A new device

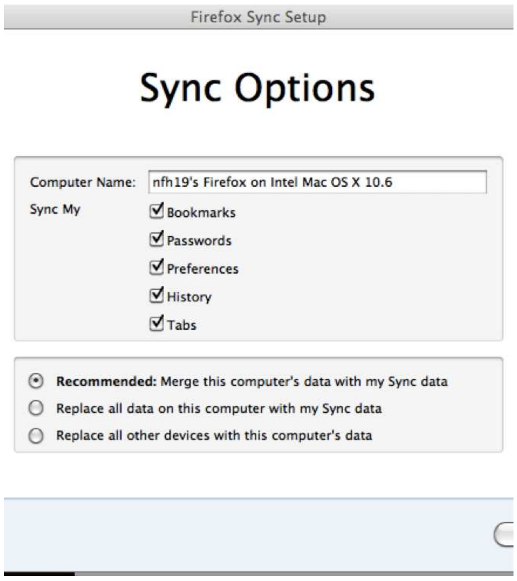
Fig. 3. Sync mechanism in Firefox 4 (beta 8 and later)

to securely transfer the sync-key between different Firefox clients. (Otherwise, the user will need to manually type in the sync-key, which can prove tedious especially on a mobile phone.) Using the J-PAKE protocol, the user reads a 12-character secret code from the new device (as shown in Figure 5) and enters it to the host device. Since the secret code is exclusively shared by the two devices, a secure end-to-end channel can be created and through this channel, the sync-key is transferred to the new device.

4 Discussion

4.1 Comparison between Firefox and Chrome

Between the two sync mechanisms, which is more appealing to users? Obviously, the Firefox sync is more secure than the Chrome's. On the other hand, many average users find the Chrome sync attractive as it is so simple and easy. It is yet unclear to what extent do users care about their privacy or whether they care enough to make a switch. In the Mozilla solution, users are in control of their data. The data is encrypted by a cryptographically strong key and only the user has access to the key. The use of J-PAKE facilitates the transfer of the sync key between devices without compromising security. However, the crypto process is not easy to understand by the common people. To many users, the sync setup in Firefox 4 happens almost like a magic. Will the Mozilla's efforts in honoring the user privacy pay out in the long term? Perhaps, only time can tell.


Firefox Sync Setup

Sync Options

Computer Name: nfh19's Firefox on Intel Mac OS X 10.6

Sync My
- ✓ Bookmarks
- ✓ Passwords
- ✓ Preferences
- ✓ History
- ✓ Tabs

- [✓ **Recommended:** Merge this computer's data with my Sync data]
- Replace all data on this computer with my Sync data
- Replace all other devices with this computer's data

Fig. 4. Sync options in Firefox 4

4.2 Outlook of PAKE

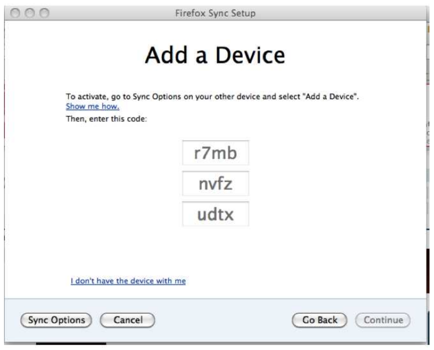
To our best knowledge, the use of J-PAKE in Firefox 4 is the first large-scale deployment of the PAKE technology. The adoption of PAKE had been greatly hampered in the past due to patent and technical issues. The obstacles are disappearing. With the rise of J-PAKE as a patent-free solution and also that the EKE patent will soon expire in October 2011 (see [6]), it looks likely that the PAKE technology will be more widely adopted in the future.

5 Conclusion

The Password Authenticated Key Exchange (PAKE) protocol is a useful cryptographic technique. In this paper, we explained how PAKE could be applied to tackle the sync problem. In particular, we described how sync works in Firefox 4, which is the first browser to adopt the PAKE technology. After over twenty years of intensive research in PAKE, the field finally starts to see its use in a large-scale practical deployment.

References

1. Feng Hao, Peter Ryan, "J-PAKE: Authenticated Key Exchange Without PKI", Springer Transactions on Computational Science, Special Issue on Security in Computing, Part II, Vol. 6480, pp. 192-206, 2010.



To activate, go to Sync Options on your other device and select "Add a Device".
Show me how.
Then, enter this code:

r7mb
nvfz
udtx

Fig. 5. Add a new sync device in Firefox 4

2. Feng Hao, Peter Ryan, "Password Authenticated Key Exchange by Juggling", Proceedings of the 16th Workshop on Security Protocols, Cambridge, April 2008.
3. Dropbox website: `http://www.dropbox.com`
4. Official SRP website: `http://srp.stanford.edu/`
5. S. Bellovin and M. Merritt, "Encrypted Key Exchange: password-based protocols secure against dictionary attacks," Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1992.
6. S. Bellovin and M. Merritt, "Cryptographic protocol for secure communications," U.S. Patent 5,241,599.
7. D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, Vol. 26, No. 5, pp. 5–26, October 1996.
8. D. Jablon, "Cryptographic methods for remote authentication," U.S. Patent 6,226,383, March 1997.
9. B. Jaspan, "Dual-workfactor Encrypted Key Exchange: efficiently preventing password chaining and dictionary attacks," Proceedings of the Sixth Annual USENIX Security Conference, pp. 43-50, July 1996.
10. IEEE P1363.2 Working Group, P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques. Draft available at `http://grouper.ieee.org/groups/1363/`.
11. Muxiang Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, Vol. 8, No. 1, pp. 63-65, January 2004.
12. F. Hao, "On small subgroup non-confinement attacks", proceedings of the 10th IEEE International Conference on Computer and Information Technology, CIT'10, pp. 1022-1025, 2010.