

# How WWPass Works

Mikhail Vysogorets

WWPass Chief Technology Officer

October 2011

## Table of Contents

---

Introduction .....	3
Safe deposit box: two key approach .....	4
PassKeys .....	6
Digital world: encryption & data dispersion .....	6
Zero knowledge policy .....	7
WWPass participants .....	8
WWPass core network structure .....	9
More on PassKeys .....	10
Password as a second authentication factor .....	10
User consent and control .....	11
Authentication ticket: how to associate Service Provider with the User .....	11
Zero knowledge: how to name & store data .....	12
Zero knowledge: I lost my PassKey (and forgot my password) .....	14
WWPass Applications: How to use WWPass technology .....	15

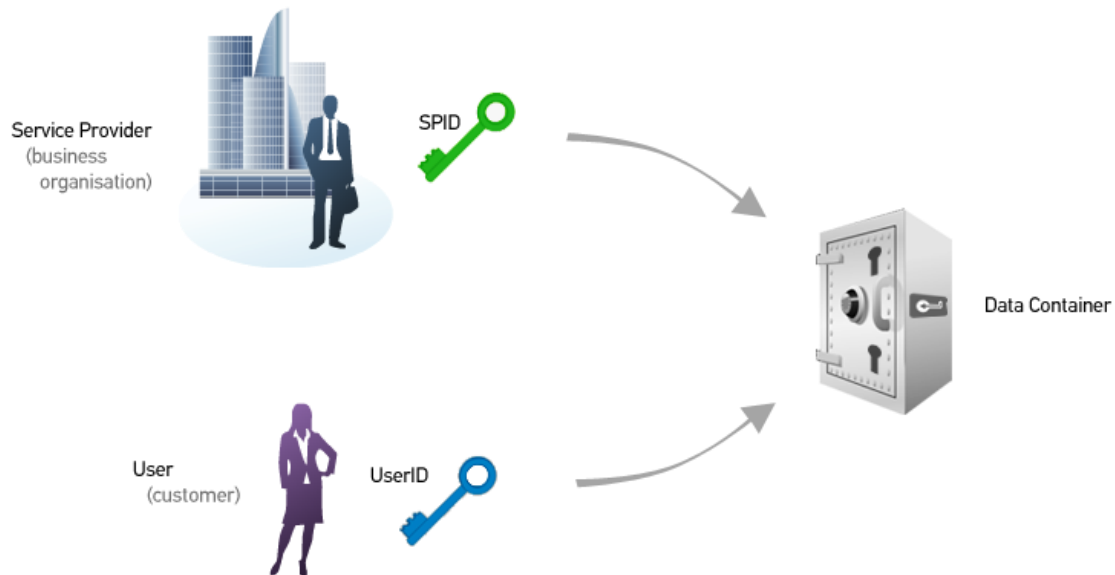
## Introduction



WWPass was founded to provide more convenient and secure authentication and data storage to both companies and end users. Currently, most of us have dozens of bank and loyalty cards, an overweight keychain, and lots and lots of Internet registrations. Every time we need to authenticate ourselves, either physically or online, we are forced to provide a card or a login/password pair. Every day we must make a difficult choice between the insecurity of a single universal password, and the inconvenience of hundreds of different cryptographically strong passwords that are impossible to remember. WWPass reconciles these security and convenience concerns.

Companies providing internal or external data services (“Service Providers”) can also benefit from WWPass technology, which allows their end-users’ private data to be stored much more securely than is currently possible, with greatly reduced insider or attacker access to sensitive information.

## Safe deposit box: two key approach



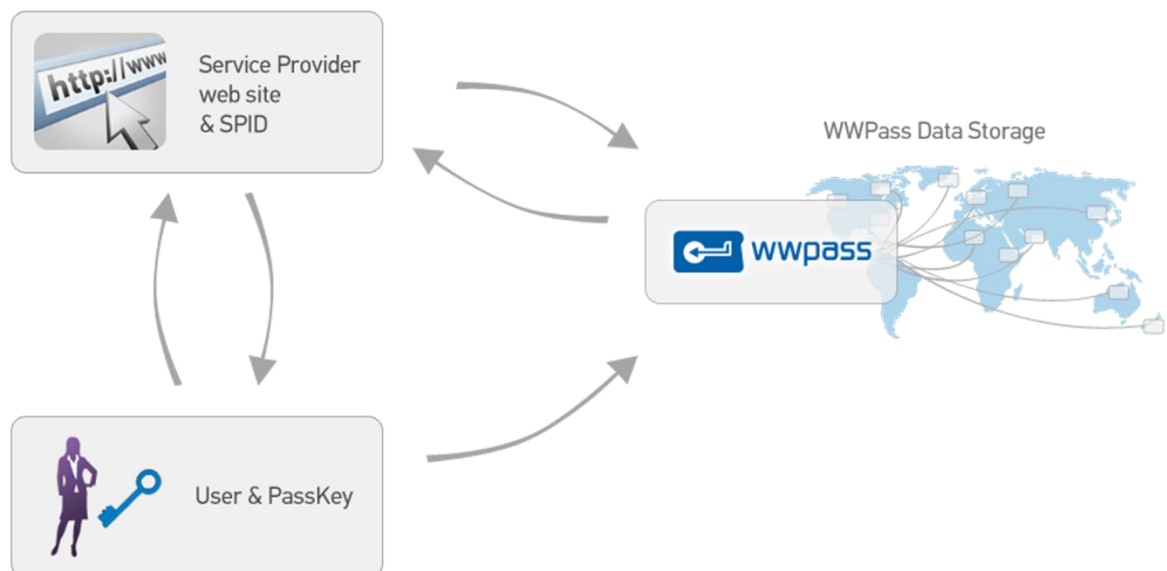
The core concept in a WWPass solution is the "Data Container", which is analogous in the non-computer world to a safe deposit box. Using a safe deposit box, a customer can store precious personal or sensitive items. When using a WWPass Data Container, a user can store valuable information in digital form. In real life, safe deposit boxes require two keys before they can be opened: the customer's key (provided by the bank), and the bank's security key. In the digital world, WWPass provides a unique digital identifier to every user (called a UserID) and another unique digital identifier to each Service Provider (called a SPID). This SPID is used by Service Providers that wish to authenticate their users.

The Service Provider may be a mail server, an on-line retail shop, a corporate web page, a kiosk, a bank, or any other application requiring authentication. WWPass provides a "safe deposit box" (Data Container) corresponding to each user registration at a particular Service Provider's web site or application (i.e. for each UserID/SPID pair). To open this box, two digital identifiers are needed: first the user's UserID and second, the Service Provider's SPID. Thus, the user's data belonging to different Service Providers never intersects and is totally independent.

For an even greater level of security, Service Providers can elect to require users to provide a second authentication factor, such as a WWPass password in addition to their UserID. This is one case where it is safe for the user to use the same password for all (WWPass-enabled) applications.

So here's how a WWPass-enabled model works for a website login:

- The User requests to be logged in to the Service Providers' web site
- The Service Provider communicates with a WWPass data center and provides its SPID
- The Service Provider asks the User to provide a UserID to WWPass
- WWPass uses both identifiers to open the corresponding Data Container and passes its content to the Service Provider
- If the user does not yet have a profile with the Service Provider application, WWPass will notify the Service Provider of the login attempt, and the Service Provider will be able to create a new Data Container to store the new user's profile.

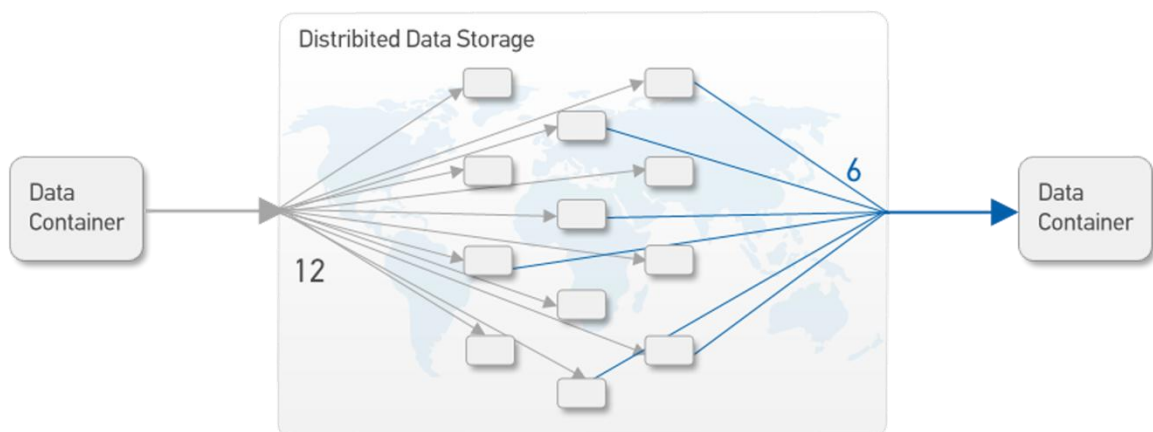


## PassKeys

In the digital world, “keys” (digital identifiers) may be copied in the same way as in real life; if someone malicious gets a replica, it is possible for the user not to notice its existence for some period of time. WWPass puts enormous effort into preventing unauthorized “key” copies. UserIDs are stored in a special secure hardware device called a PassKey. These PassKeys are constructed with well-known "smart card" technology that is widely used in chip cards and mobile phone SIM cards. One PassKey and its associated password can actually replace many existing cards, keys and login-password pairs for a WWPass-enabled User.

## Digital world: encryption & data dispersion

Compared to the real world, the digital world provides additional security capabilities. The first and most advanced is encryption. WWPass encrypts all data; every Data Container is encrypted with its individual cipher key. Thus not only is all the data stored by WWPass unreadable, but there is no single secret "cipher key" which, when revealed, decodes all WWPass information.



But WWPass goes further. In order to survive a single storage node breach, WWPass distributes the Data Container content over multiple geographic locations. Every Data Container's content is converted into twelve unrecognizable pieces and stored at twelve different data centers. This is accomplished using the Reed-Solomon algorithm that was originally designed for highly robust data transmission. In the WWPass implementation of Reed-Solomon, dispersed data can be restored so long as any six of the twelve pieces are available. At the same time, if a malicious entity managed to gain access to less than six pieces of data, it would be impossible to restore any of the original data.

As previously noted, WWPass uses the Reed-Solomon redundancy code (6,12) to implement this feature. The idea to use Reed-Solomon for data dispersion was first proposed by M. Rabin<sup>1</sup>, and has subsequently been implemented in RAID6 storage arrays and Tahoe local networks.

## Zero knowledge policy

Zero knowledge is one of the main principles of the WWPass architecture. This principle means that all users are anonymous to WWPass, which has no knowledge of any data stored in Data Containers. WWPass asserts to its Service Providers that, "when the owner of this particular PassKey visited your web site previously, you (the Service Provider) asked WWPass to store the following bytes." Furthermore, IDs are not retained in the WWPass Core system, so it is impossible to find out the owner of a particular Data Container without a PassKey. Generally speaking this "zero knowledge" principle is the ultimate solution to insider security breaches and leaks (also known as back doors).

As a side effect, WWPass cannot do much when a user loses his/her hardware device. Therefore, we empower the user to perform all PassKey operations: revoke

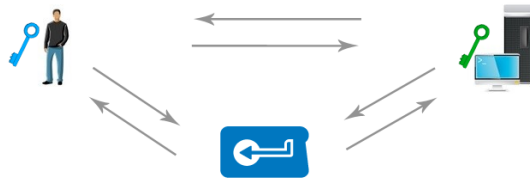
---

<sup>1</sup> "Efficient dispersal of information for security, load balancing, and fault tolerance", <http://dl.acm.org/citation.cfm?id=62050>

lost PassKey, create new PassKeys etc. Moreover, WWPass provides the user with two types of devices: the first is called a "PassKey" and is intended for everyday use; the second, called a "Service Key", performs only management tasks.

## WWPass participants

WWPass participants are shown on the following picture:

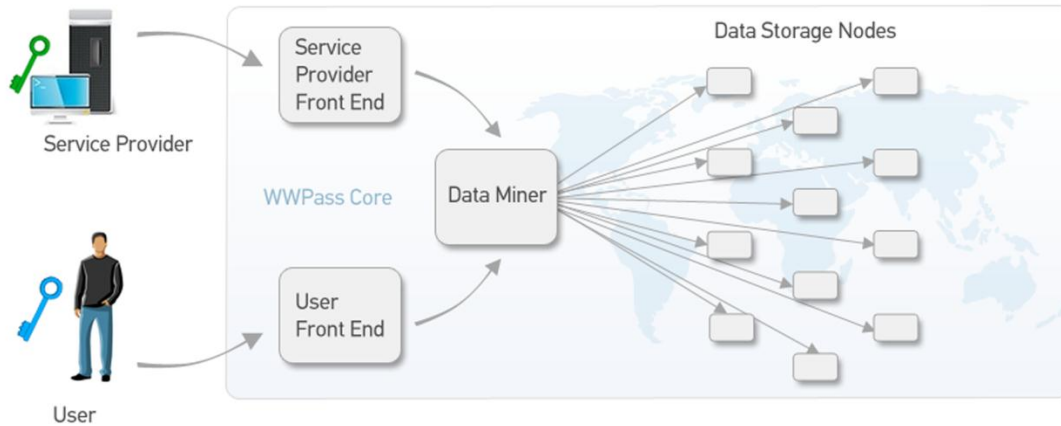


There are three parties in the WWPass model: A Service Provider with an Application or web service, a User with a user terminal, and the WWPass core network.

- A "User Terminal" can be a person's home or office desktop/laptop, a retail store checkout, kiosk, etc. To start a WWPass transaction, the User connects his/her PassKey to the User Terminal.
- The WWPass core network provides mutual authentication service between all participants and stores the user's private data in a secure, dispersed Data Container.
- The term "Service Provider Application" actually means application servers or web servers. WWPass.com and id.wwpass.com are examples of Service Providers; these sites are not a part of the WWPass Core network, they are examples of working Service Provider Applications.

## WwPass core network structure

Digging deeper into the WwPass core network, the following components can be found:



- Service Provider Front-ends (SPFE) and User Front-ends (UserFE) are network entities responsible for Service Provider and User authentication. User passwords are also checked here. The SPFE transmits the Data Container contents between Service Providers and WwPass Storage.
- Data Miners communicate with storage nodes and disperse Data Containers on write commands, and assemble them back on read commands. Data miners are responsible for preserving data integrity, and for error detection and correction.
- Storage Nodes are a set of geographically dispersed servers that store data.

It is important to note that there may be multiple Front Ends and Data Miners running concurrently. Thus it is possible to implement a reliable system without any single point of failure.

## More on PassKeys

Pass Keys are based on well-known "smart card" technology that is widely used in bank chip cards and mobile phone SIMs. To be precise, we use so-called Java Cards. This technology is able to perform most crypto operations, such as symmetric and asymmetric cipher, digital signature, and message integrity check. Java Cards are very flexible and can be easily adapted to any kind of protocol or technology. Java Cards are cryptographically strong; it is virtually impossible for a Black Hat to read their memory and to steal crypto keys and other secrets.

Java Cards are an industrial standard, so WWPass can use any Java Card available on the market as the PassKey. This makes WWPass technology inexpensive and robust while providing the highest level of data protection. When used as bank cards or SIMs, smart cards are usually mounted on modules with eight gold-plated contacts and then implanted into a piece of plastic. Some cards may use a so-called "contactless" interface. An example is transportation cards, which are used widely all around the world.

Due to the well-established industry status of Java Cards, WWPass can provide its PassKeys in multiple form factors and interfaces, including bare plastic smart cards, USB fobs, and smartphones.

## Password as a second authentication factor

Authentication strength is usually improved with additional factors. Depending on the situation, a Service Provider could ask the user to provide a password along with a PassKey. An important difference is that the user inputs the password only to WWPass, not to every Service Provider. Thus it is not necessary to create and remember many passwords. This password approach is similar in functionality to a bank card PIN; the PIN is associated with a card (i.e. – a PassKey), not with an ATM (i.e. – an application). According to the zero-knowledge principle, WWPass cannot

help if a user forgets their password. However, with both a PassKey and a Service Key a User can reset and change their password.

Technically, WWPass employs the SRP (Secure Remote Password) protocol.<sup>2</sup> The protocol is now widely used for strong network authentication and is described in RFC2945 and is ISO - standardized (ISO/IEC 11770-4).

## User consent and control

According to Kim Cameron's paper "The Laws of Identity"<sup>3</sup>, "Technical identity systems must only reveal information identifying a user with the user's consent."

This means that every act of user identification should be done under the full consent and control of the user. WWPass meets this requirement by providing the user with a confirmation button either in the form of a hardware button on a physical PassKey, or a touchscreen button on a smartphone PassKey. To take it one step further, "should the user decide to supply identity information, there must be no doubt that it goes to the right place". WWPass meets this requirement by displaying—either on the User Terminal or a smartphone display—the validated name for the Service Provider that is requesting authentication.

## Authentication ticket: how to associate Service Provider with the User

In order to provide a proper Data Container, WWPass needs to associate a particular user with a specific Service Provider. To do this, a temporary transaction identifier called a "ticket" is employed. The term itself is taken from Kerberos, a widespread authentication protocol. Yet a WWPass ticket differs from a Kerberos

---

<sup>2</sup> <http://srp.stanford.edu>

<sup>3</sup> <http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf>

ticket in many aspects. Roughly speaking, a WWPass ticket combines a “nonce” (number used once) and a Service Provider name.

A ticket is issued by the WWPass core service upon request by the Service Provider. The Service Provider transmits the ticket to the user. This ticket is loaded into the WWPass PassKey. The user sees the Service Provider name, which piggybacks on the ticket, and confirms the transaction. Next, the PassKey starts the authentication process with the WWPass core network. Upon success, a ciphered communication channel is created between the PassKey and the WWPass User Front End. The ticket and the UserID are transmitted secretly via this channel back to WWPass. Now WWPass can determine the Service Provider ID (SPID), which corresponds to the requesting Service Provider.

Having both identifiers, the SPID and the UserID, WWPass calculates the Data Container address and its encryption key. With the address available the data is ready to be transmitted to the Service Provider. WWPass authenticates PassKeys according to the GlobalPlatform 2.2.1 standard (Secure Channel version 2).<sup>4</sup>

## **Zero knowledge: how to name & store data**

Only when WWPass has both the UserID and the SPID can it store and retrieve information from the Data Container. A straightforward approach would be to use a concatenation of the UserID and the SPID as a container identifier (or filename, or key) and to store the Service Provider data as is. If WWPass could guard its storage media from the outside world, this solution might be safe. However if the storage media itself can be subject to criminal investigation, lots of information could be harvested. Existing UserIDs could be extracted from the container identifiers, as well as their combinations with SPIDs. This would allow user activities at various web sites to be tracked.

---

<sup>4</sup> <http://www.globalplatform.org>

WWPass could use hash functions, but unfortunately they do not guarantee uniqueness. What is needed is an “Injective” transformation. (An injective function is a function that preserves distinctness; it never maps distinct elements of its domain to the same element of its co-domain. In other words, every element of the function's co-domain is mapped to, by *at most* one element of its domain). To protect IDs in a Data Container, WWPass uses a transformation for identifiers; this is in fact a “one-way injective function”. This means WWPass applies a specific function to the UserID/ SPID combination to create a unique container identifier. “One-way” means that it is virtually impossible to “reverse” the transformation and get back the UserID and SPID.

At least two functions are known to have this property. The first is a power function modulo prime number (used in e.g. Diffie-Hellman key exchange algorithm<sup>5</sup>). Another solution is to use an RSA algorithm for public-key encryption. So long as the private key is intentionally destroyed, this encryption becomes one-way.<sup>6</sup> WWPass uses the RSA algorithm.

Both functions mentioned work in modulo arithmetic, typically with quite big numbers (1024 or 2048 bits long). Treated as bit strings, those numbers may hold concatenations of the UserID and SPID. Spare bits allow WWPass to use additional parameters – e.g. container names. Thus a large number of named Data Containers for each User/Service Provider combination may be provisioned.

Encryption is highly desirable to prevent the malicious interrogation of storage media. While Reed-Solomon dispersion can be considered a method of keeping data secret (see Adi Shamir’s article “How to share a secret”:  
<http://securespeech.cs.cmu.edu/reports/shamirturing.pdf>), WWPass nonetheless encrypts all Data Containers before dispersion. Furthermore, each data container is ciphered with its own encryption key. Moreover, employing an “almost unique”,

---

<sup>5</sup> <http://www.ietf.org/rfc/rfc2631.txt>

<sup>6</sup> [http://www.di-mgt.com.au/rsa\\_alg.html](http://www.di-mgt.com.au/rsa_alg.html)

unpredictable cipher key enables the use of any hash function, e.g.  $K = \text{SHA1}(\text{UserID}, \text{SPID})$ . Finally, WWPass applies a standard symmetric cipher algorithm to the original data.

WWPass can check integrity and correct errors without any knowledge of customer data. Iterating over available cryptic names of containers makes it possible to determine if a chunk has been lost on a particular storage node, and to restore it properly using Reed-Solomon codes.

To summarize how WWPass handles customer data: the container name and content are ciphered; keys are unique for each container and become known to WWPass only during the actual transaction, i.e. - when the SPID and UserID are available; since IDs are not available to WWPass outside the transaction, it is impossible for WWPass to calculate the owner of any particular Data Container or to decipher its content.

## **Zero knowledge: I lost my PassKey (and forgot my password)**

User anonymity comes at a price: WWPass itself can do nothing in cases where a user loses their PassKey or forgets a password. If there were no way to restore the PassKey, all data contained by WWPass would be lost. To mitigate this situation, WWPass empowers users to control their PassKeys, with the ability to block, name, create, and replace them. Users are responsible for the safe keeping of their WWPass credentials.

WWPass provides two types of Keys: a “PassKey” for everyday needs, and a “Service Key” that is bound to the user account. The Service Key cannot be used for e-shopping or corporate site login. However, it is the Service Key that can revoke/disable a lost PassKey and provide the necessary data to create a new PassKey. If the user forgets a password, the old password cannot be recovered

since WwPass operates only with derivative values. However, using the Service Key and the PassKey, the user can reset their password for use across all WwPass-enabled Service Providers. WwPass customers are presented with two Service Keys as part of their KeySet package. Users are encouraged to store one Service Key with a third party, or in a safe recoverable location such as a physical safe or a bank safe deposit box.

## WwPass Applications: How to use WwPass technology



WwPass concentrates its efforts on core security development and prefers third-party Service Providers and developers to implement WwPass support into their applications. WwPass provides a fully documented interaction between the Service Provider application and the WwPass Front End. These documents are available in conjunction with PHP and Python libraries, along with examples, in a WwPass Software Development Kit.

To prove the WwPass concepts and to demonstrate the technology, WwPass has implemented services as well as modules for a wide range of web-based frameworks (WordPress, Zen Cart, Magento). These demonstration modules show that some common off-the-shelf technologies can be made more secure and convenient. See, for example, the WwPass OpenID application ([id.wwpass.com](http://id.wwpass.com)).

Users may also store their web login-password pairs in the WWPass web based password storage application (pws.wwpass.com).

WWPass's flagship application is called PSS (Private Secure Storage). PSS is a cloud storage system based on smart card cryptography. It provides a high level of security and does not use any Web technologies to upload/download user files. The PSS application is available for the Windows and MAC operating systems. WWPass also provides a convenient way to sign and encrypt/decrypt e-mail, connect to VPNs, etc. with a PKCS11 library.<sup>7</sup> Third party developers may use WWPass APIs and libraries to create User Terminal applications with WWPass authentication and secure storage.

---

<sup>7</sup> <http://www.rsa.com/rsalabs/node.asp?id=2133>