OneID – An architectural overview

Jim Fenton
November 1, 2012

Introduction

OneID is an identity management technology that takes a fresh look at the way that users authenticate and manage their identities on the Internet. Since it is a radical departure from most previous technologies in this area, this document gives a brief overview of how various OneID operations function and the rationale for many of the design decisions. This document assumes that the reader has a technical background and is familiar with OneID from a functional standpoint.

This document begins with some aspects of the OneID infrastructure, namely key management and password/personal identification number (PIN) verification, and then continues with descriptions of the primary OneID functional processes.

OneID commonly uses some terms that may not be familiar to the reader:

- Relying party (RP), website, or site – The service being accessed by the user
- Access device (AD) or device – An agent, typically a browser, on which the user initiates a transaction
- Control device (CD) or OneID Remote app – An agent on which the user confirms a transaction
- Repository (repo) – A cloud service operated by OneID or partners that holds encrypted user information, enforces user policy, and cosigns transactions
- Out-of-band – confirmation of a sign in or transaction through an independent device such as the OneID Remote app.

For consistency with internal naming conventions, this document uses somewhat different terminology than OneID user-facing documentation.

Key Management

OneID has a more complex key management structure than most systems of this sort to meet its goal of supporting arm's-length relationships among the three classes of agents representing the user: their [access] device, their OneID Remote app, and the repository they use. To keep the number of keys manageable, OneID makes extensive use of key derivation functions to create the keys from a smaller number of master keys.

In order to provide consistent user signatures independent of which devices the user may be using, each class of agents has a *master key* that is generated when the

user's account is created, or in the case of the OneID Remote app, when the first Remote app is added to their account. These master keys are securely transferred to new devices when the user adds them to their account.

In order to allow users to individually authorize their devices and mobile apps, there is also an individual device identifier that is checked by the repository. The repository will block transactions from proceeding if the user device is found to be not authorized or has been temporarily locked by the user. The repository can also enforce device-specific security requirements, such as approval of a OneID Remote, if desired by the user.

One of the elements of OneID's privacy framework is the use of *directed identity* to identify the user at sites where they use their OneID. This prevents sites from correlating the user's activities based on their OneID. Although in many cases users will release other information that may identify them, this capability is important to preserve pseudonymity in cases where this is required. To achieve this, OneID securely derives new keys from the respective master keys and the domain of the site the user is accessing. These keys are consistent when the same user visits the same site, but cannot be used to correlate with activity at other sites.

Password and PIN management

OneID uses two authentication factors. The first is the user's possession of a device with stored keying information. The second is a memorized password (used on access devices) and PIN (used on OneID Remote apps). Password and PIN management takes advantage of keying information stored in the user's endpoints, enabling secure verification of passwords and PINs in a way that isn't subject to dictionary attack. Passwords and PINs never leave the device on which the user enters them.

Passwords and PINs are verified by deriving a private key from the password or PIN entered by the user combined with a secret salt derived from the device's master key. The user's device signs a challenge nonce from the repository using this key, and the repository verifies this signature using the corresponding public key that it has stored. Since the repo never has knowledge of the 128-bit salt value, dictionary attacks are not feasible. The repository signature attests to, among other things, the successful verification of the PIN and/or password as required. The repository also enforces limits on the rate of incorrect password or PIN verifications it will perform, to protect against an attacker with access to one of those devices.

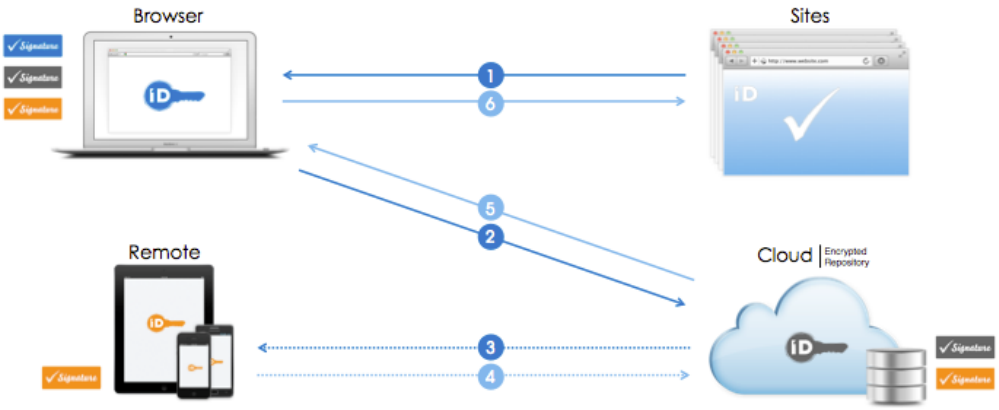
Attribute Management

User attributes—currently self-asserted information from the user, such as their name, address, and credit card information—are encrypted at the user's device and

are stored in encrypted form in the repository. As additional protection on the nature of the information stored in the repo, the AD also deterministically encrypts (using a fixed initialization vector) the names of attributes and the names of sites with which the user has authorized the attributes to be shared. When an attribute is to be retrieved, the AD encrypts the attribute and relying party name deterministically, and sends those to the repo for retrieval. The repo also determines when it is necessary for the AD to prompt the user for permission to share an attribute that has not been shared with that site previously.

Authentication

User authentication starts with a challenge nonce generated by the relying party. The user's device (browser), the repository, and optionally one of the user's OneID Remote apps then generate signatures upon their own copies of the nonce. The signed nonces are returned to the RP via an SSL callback, along with attribute data requested by the RP and some information confirming the RP identity and the type of authentication performed.



Let's look at this process in more detail.

1. The user device accesses a OneID-enabled website which returns a digital challenge (known as a "nonce") and requests that the user authenticate by providing signatures from their device and their OneID repository. The site can also request a third signature from the user's OneID remote, can specify entry of a PIN or password as additional security, and can request that the user share some information (*attributes*) stored in their repository as part of the authentication process. It also supplies a callback URL to be used to return the authentication signatures.
2. The user device passes the challenge, proof of the identity of the specific device, and information about the request (encrypted for privacy reasons) to their OneID credential and data storage repository. If the website has

requested user attributes, it encrypts the attribute request and sends it to the repository as well.

3. If either the request or the user's preferences require participation of a OneID Remote, the repository sends the challenge and the encrypted description of the transaction being approved to the user's OneID Remote app.
4. If required, the OneID Remote app decrypts the request using keying information it has stored and displays it to the user. Depending on the security requirements of the user and the website, it may prompt the user to enter a PIN. If the user consents, the OneID Remote app will sign the challenge using a private key known only to Remote apps. It sends the signed challenge, a signature representing the identity of the individual OneID Remote app, and a cryptographic verifier for the PIN (not the PIN itself) back to the repository.
5. If all of the security requirements specified by the user and the website are satisfied, the PIN if required was entered correctly, and all of the user devices have proven to the repository that they are authorized devices, the repository will sign the challenge and return the signed challenges to the user's device. If attributes were requested, the repository retrieves those encrypted values and whether they have previously been released to this website, and includes those in the response. If attributes are being provided to this website that have not been previously released, the user's device obtains user consent. If authorized, the user device decrypts the attributes.
6. The user's device signs the challenge using a private key derived from the device's master key and the site name and returns all the signed challenges and decrypted attributes to the website via the callback URL provided in step 1. The website verifies all signatures are correct, grants access to the user, and makes use of whatever attributes may have been provided. The website never communicates directly with the user's repository, placing the user directly in control and limiting the information available to both the website and the repository.

Adding devices

In order to make it possible for a user to use their OneID on more than one device, OneID has a process known as device addition to securely transfer keying information from one device to another and capabilities to manage devices through the OneID Control Panel. In order to keep this process as easy to use as possible while maintaining security, this is facilitated through the use of QR codes (two-dimensional barcodes) that are scanned by the user's OneID Remote app using that device's camera.

The device to be added to the user's account initiates this process by generating and displaying a QR code that is used to establish the connection between the devices

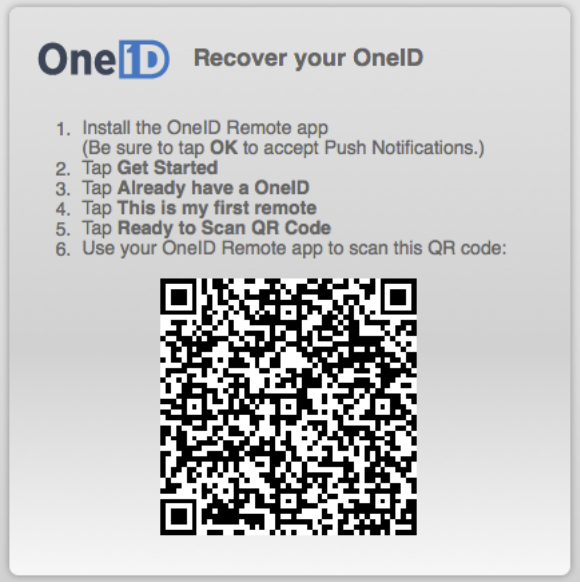
and to communicate a short-term secret that can be used to securely transfer the keying information. The user must also provide the password or PIN as appropriate for the device being added. The user's OneID Remote app scans the code and, if the repository correctly verifies the PIN or password, facilitates the transfer of encrypted keying information to the new device (the repository never has access to the user devices' keying information).

QR code displayed on a device being added

A special case of this process is the addition of the very first OneID Remote to a user's account. In this case, the flow is reversed: the new OneID Remote scans a QR code displayed on the user's browser to associate it with the account. The new OneID Remote generates its own master key and PIN verifier, and calculates the necessary public keys and sends them to the repository for future use. Once the first OneID Remote is added to the user's OneID account, all future addition of devices and additional OneID Remotes is approved by one of the OneID Remote apps already associated with the user's account.

Account Recovery

OneID gives the user the ability to create an account recovery URL that can be rendered as a QR code for recovery of their account. The URL contains a key that is used to encrypt a copy of the user's device secrets for storage in the repository. The user can store this URL/QR code in any manner they wish: they can print out the QR code and store it in a safe place, they can send the URL in an email to themselves (which, of course, limits the security of the OneID account to that of their email account), or they can store it electronically in a manner of their choosing (cloud storage service, USB memory stick, etc.).



Sample OneID Recovery QR code

The recovery code URL references a service that renders the QR code locally in the user's browser. After scanning the code and prompting the user for their PIN code, the Remote app receives the necessary keying information and is added to the user's account. The user can then manage the other devices on their OneID account, including adding new devices and removing any devices that may have been lost or stolen.

