One-time password

From Wikipedia, the free encyclopedia

A **one-time password** (**OTP**) is a password that is valid for only one login session or transaction, on a computer system or other digital device. OTPs avoid a number of shortcomings that are associated with traditional (static) password-based authentication; a number of implementations also incorporate two factor authentication by ensuring that the one-time password requires access to *something a person has* (such as a small keyring fob device with the OTP calculator built into it, or a smartcard or specific cellphone) as well as *something a person knows* (such as a PIN).

The most important advantage that is addressed by OTPs is that, in contrast to static passwords, they are not vulnerable to replay attacks. This means that a potential intruder who manages to record an OTP that was already used to log in to a service or to conduct a transaction will not be able to abuse it, since it will no longer be valid. A second major advantage is that a user who uses the same (or similar) password for multiple systems, is not made vulnerable on all of them, if the password for one of these is gained by an attacker. A number of OTP systems also aim to ensure that a session cannot easily be intercepted or impersonated without knowledge of unpredictable data created during the *previous* session, thus reducing the attack surface further.

OTPs have been discussed as a possible replacement for, as well as enhancer to, traditional passwords. On the downside, OTPs are difficult for human beings to memorize. Therefore, they require additional technology to work.

Contents

- 1 How OTPs are generated and distributed
- 2 Methods of generating the OTP
  - 2.1 Time-synchronized
  - 2.2 Mathematical algorithms
- 3 Methods of delivering the OTP
  - 3.1 Text messaging
  - 3.2 Mobile phones
  - 3.3 Proprietary tokens
  - 3.4 Web-based methods
  - 3.5 Hardcopy
- 4 Comparison of technologies
  - 4.1 Comparison of OTP implementations
  - 4.2 OTPs versus other methods of securing data
  - 4.3 Related technologies
- 5 Standardization
- 6 See also
- 7 References

How OTPs are generated and distributed

OTP generation algorithms typically make use of pseudorandomness or randomness, making prediction of successor OTPs by an attacker difficult, and also hash functions, which can be used to derive a value but are hard to reverse and therefore difficult for an attacker to obtain the data that was used for the hash. This is necessary because otherwise it would be easy to predict future OTPs by observing previous ones. Concrete OTP algorithms vary greatly in their details. Various approaches for the generation of OTPs are listed below:

- Based on **time-synchronization** between the authentication server and the client providing the password (OTPs are valid only for a short period of time)
- Using a mathematical **algorithm** to generate a new password **based on the previous password** (OTPs are effectively a chain and must be used in a predefined order).
- Using a mathematical **algorithm** where the new password is **based on a challenge** (e.g., a random number chosen by the authentication server or transaction details) and/or a counter.

There are also different ways to make the user aware of the next OTP to use. Some systems use special electronic security tokens that the user carries and that generate OTPs and show them using a small display. Other systems consist of software that runs on the user's mobile phone. Yet other systems generate OTPs on the server-side and send them to the user using an out-of-band channel such as SMS messaging. Finally, in some systems, OTPs are printed on paper that the user is required to carry.

Methods of generating the OTP

Time-synchronized

A time-synchronized OTP is usually related to a piece of hardware called a security token (e.g., each user is given a personal token that generates a one-time password). It might look like a small calculator or a keychain charm, with an LCD that shows a number that changes occasionally. Inside the token is an accurate clock that has been synchronized with the clock on the proprietary authentication server. On these OTP systems, time is an important part of the password algorithm, since the generation of new passwords is based on the current time rather than, or in addition to, the previous password or a secret key. This token may be a proprietary device, or a mobile phone or similar mobile device which runs software that is proprietary, freeware, or open-source. An example of time-synchronized OTP standard is Time-based One-time Password Algorithm (TOTP).

All of the methods of *delivering* the OTP below may use time-synchronization instead of algorithms.

Mathematical algorithms

Each new OTP may be created from the past OTPs used. An example of this type of algorithm, credited to Leslie Lamport, uses a one-way function (call it f). This one-time password system works as follows:

1. A seed (starting value) s is chosen.
2. A hash function $f(s)$ is applied repeatedly (for example, 1000 times) to the seed, giving a value of: $f(f(f($ .... $f(s)$ ....$)))$. This value, which we will call $f^{1000}(s)$ is stored on the target system.

3. The user's first login uses a password p derived by applying f 999 times to the seed, that is, $f^{999}(s)$. The target system can authenticate that this is the correct password, because $f(p)$ is $f^{1000}(s)$, which is the value stored. The value stored is then replaced by p and the user is allowed to log in.
4. The next login, must be accompanied by $f^{998}(s)$. Again, this can be validated because hashing it gives $f^{999}(s)$ which is p, the value stored after the previous login. Again, the new value replaces p and the user is authenticated.
5. This can be repeated another 997 times, each time the password will be f applied one fewer times, and is validated by checking that when hashed, it gives the value stored during the previous login. Hash functions are designed to be extremely hard to reverse, therefore an attacker would need to know the initial seed s to calculate the possible passwords, while the computer system can confirm the password on any given occasion is valid by checking that, when hashed, it gives the value previously used for login. If an indefinite series of passwords is wanted, a new seed value can be chosen after the set for s is exhausted.

To get the next password in the series from the previous passwords, one needs to find a way of calculating the inverse function f^{-1}. Since f was chosen to be one-way, this is extremely difficult to do. If f is a cryptographic hash function, which is generally the case, it is (so far as is known) a computationally infeasible task. An intruder who happens to see a one-time password may have access for one time period or login, but it becomes useless once that period expires. The S/KEY one-time password system and its derivative OTP are based on Lamport's scheme.

In some mathematical algorithm schemes, it is possible for the user to provide the server with a static key for use as an encryption key, by only sending a one-time password.[1]

The use of challenge-response one-time passwords requires a user to provide a response to a challenge. For example, this can be done by inputting the value that the token has generated into the token itself. To avoid duplicates, an additional counter is usually involved, so if one happens to get the same challenge twice, this still results in different one-time passwords. However, the computation does not usually involve the previous one-time password; that is, usually this or another algorithm is used, rather than using both algorithms.

The methods of delivering the OTP which are *token-based* may use either of these types of algorithm instead of time-synchronization.

Methods of delivering the OTP

Text messaging

A common technology used for the delivery of OTPs is text messaging. Because text messaging is a ubiquitous communication channel, being directly available in nearly all mobile handsets and, through text-to-speech conversion, to any mobile or landline telephone, text messaging has a great potential to reach all consumers with a low total cost to implement. However, the cost of text messaging for each OTP may not be acceptable to some users. OTP over text messaging may be encrypted using an A5/x standard, which several hacking groups report can be successfully decrypted within minutes or seconds,[2][3][4][5] or the OTP over SMS might not be encrypted by one's service-provider at all. In addition to threats from hackers, the mobile phone operator becomes part of the trust chain. In the case of roaming, more than a single mobile phone operator has to be trusted. Anyone using this information may mount a man-in-the-middle attack.

In 2011, Google has started offering OTP to mobile and landline phones for all Google accounts.[6] The user can receive the OTP either as a text message or via an automated call using text-to-speech conversion. In case none of the user's registered phones is accessible, the user can even use one of a set of (up to 10) previously generated one-time backup codes as a secondary authorization factor in place of the dynamically generated OTP, after signing in with their account password.

Mobile phones

A mobile phone keeps costs low because a large customer-base already owns a mobile phone for purposes other than generating OTPs. The computing power and storage required for OTPs is usually insignificant compared to that which modern smartphones typically use. Mobile phones additionally support any number of tokens within one installation of the application, allowing a user the ability to authenticate to multiple resources from one device. This solution also provides model-specific applications to the user's mobile.

ActivID Soft Token from HID Global security tokens.

Proprietary tokens

EMV is starting to use a challenge-response algorithm (called "Chip Authentication Program") for credit cards in Europe. On the other hand, in access control for computer networks, RSA Security's SecurID is one example of a time-synchronization type of token or HID Global. Like all tokens, these may be lost, damaged, or stolen; additionally there is an inconvenience as batteries die, especially for tokens without a recharging facility or a non-replaceable battery. A variant of the proprietary token was proposed by RSA in 2006 and was described as "ubiquitous authentication", in which RSA would partner with manufacturers to add physical SecurID chips to devices such as mobile phones.

ActivID Flexi Token from HID Global security tokens.

Recently, it has become possible to take the electronic components associated with regular keyfob OTP tokens and embed them in a credit card form factor. However, the thinness of the cards, at 0.79mm to 0.84mm thick, prevents standard components or batteries from being used. Special polymer-based batteries must be used which have a much lower battery life than coin (button) cells. Semiconductor components must not only be very flat but must minimise power used in standby and when operating.

Yubico offers a small USB token with an embedded chip that creates an OTP when a key is pressed and simulates a keyboard to facilitate easily entering a long password.[7] Since it is a USB device it avoids the inconvenience of battery replacement.

RSA SecurID security tokens.

A new version of this technology has been developed that embeds a keypad into a payment card of standard size and thickness. The card has an embedded keypad, display, microprocessor and proximity chip.

Web-based methods

Authentication-as-a-service providers offer various web-based methods for delivering one-time passwords without the need for tokens. One such method relies on the user's ability to recognize pre-chosen categories from a randomly generated grid of pictures. When first registering on a website, the user chooses several secret categories of things; such as dogs, cars, boats and flowers. Each time the user logs into the website they are presented with a randomly generated grid of picalphanumeric character overlaid on it. The user looks for the pictures that fit their pre-chosen categories and enters the associated alphanumeric characters to form a one-time access code.[8][9]

Hardcopy

In some countries' online banking, the bank sends to the user a numbered list of OTPs that are printed on paper. Other banks send plastic cards with actual OTPs obscured by a layer that the user has to scratch off to reveal a numbered OTP. For every online transaction, the user is required to enter a specific OTP from that list. Some systems ask for the numbered OTPs sequentially, others pseudorandomly chose an OTP to be entered. In Germany and many other countries like Austria and Brazil,[10] those OTPs are typically called TANs (for 'transaction authentication numbers'). Some banks even dispatch such TANs to the user's mobile phone via SMS, in which case they are called mTANs (for 'mobile TANs').

Welcome, jbarkley. Enter OTP 4:

Paper-based OTP web-site login

Comparison of technologies

Comparison of OTP implementations

The cheapest OTP solutions are those that *deliver* OTPs on paper, and those that *generate* OTPs on an existing device, without the costs associated with (re-)issuing proprietary electronic security tokens and SMS messaging.

For systems that rely on electronic tokens, algorithm-based OTP generators must cope with the situation where a token drifts out-of-sync with its server if the system requires the OTP to be entered by a deadline. This leads to an additional development cost. Time-synchronized systems, on the other hand, avoid this at the expense of having to maintain a clock in the electronic tokens (and an offset value to account for clock drift). Whether or not OTPs are time-synchronized is basically irrelevant for the degree of vulnerability, but it avoids a need to re-enter passwords if the server is expecting the last or next code that the token should be having because the server and token have drifted out-of-sync.

Use of an existing mobile device avoids the need to obtain and carry an additional OTP generator. The battery may be recharged; as of 2011 most small card devices do not have rechargeable, or indeed replaceable, batteries. However, most proprietary tokens have tamper-proof features.

OTPs versus other methods of securing data

One-time passwords are vulnerable to social engineering attacks in which phishers steal OTPs by tricking customers into providing one or more OTPs that they used in the past. In late 2005 customers of a Swedish bank were tricked into giving up their one-time passwords.[11] In 2006 this type of attack was used on customers of a US bank.[12] Even time-synchronized OTPs are vulnerable to phishing, by two methods: The password may be used as quickly by the attacker as the legitimate user, if the attacker can get the OTP in plaintext quickly enough. The other type of attack—which may be defeated by OTP systems implementing the hash chain as discussed above—is for the phisher to use the information gained (*past* OTP codes which are no longer valid) by this social-engineering method to predict what OTP codes will be used in the *future*. For example, an OTP password-generator that is pseudo-random rather than truly random might or might not be able to be compromised, because pseudo-random numbers are often predictable once one has the *past* OTP codes. An OTP system can only use truly random OTPs if the OTP is generated by the authenticator and transmitted (presumably out-of-band) to the user; otherwise, the OTP must be independently generated by each party, necessitating a repeatable, and therefore merely pseudo-random, algorithm.

Although OTPs are in some ways more secure than a static memorized password, users of OTP systems are still vulnerable to man-in-the-middle attacks. OTPs should therefore not be disclosed to any third parties, and using an OTP as one layer in layered security is safer than using OTP alone; one way to implement layered security is to use an OTP in combination with a password that is *memorized* by the user (and never *transmitted* to the user, as OTPs often are). An advantage to using layered security is that a single sign-on combined with one master password or password manager becomes safer than using only 1 layer of security during the sign-on, and thus the inconvenience of password fatigue is avoided if one usually has long sessions with many passwords that would need to be entered mid-session (to open different documents, websites, and applications); however, the disadvantage of using many forms of security all at once during a single sign-on is that one has the inconvenience of more security precautions during every login—even if one is logging in only for a brief usage of the computer to access information or an application that doesn't require as much security as some other top-secret items that computer is used for. See also Related technologies, below.

Related technologies

More often than not, one-time passwords are an embodiment of two-factor authentication (2FA) or (T-FA). T-FA is a form of layered security where it is unlikely that both layers would be disabled by someone using only one type of attack. Some single sign-on solutions make use of one-time passwords. One-time password technology is often used with a security token.