

# Keybuster Keys Security

## Introduction

Keybuster does not provide Public Key Infrastructure (PKI) security, but helps developers and other entities to protect their intellectual work from piracy and deny efficiently access to unauthorized users.

Before Keybuster, a developer had to protect their software using one of the multiple tools to wrap software protection around an existing software, and left them always in the doubt if it was good enough, although there are some tools which provide a pretty good security even against professional hackers.

After wrapping, many software ceased to function properly, in particular ActiveX components for Windows, and Dynamic Link Libraries developed with different tools than the world's leaders proposed.

Also, most protection tools are platform orientated, and an universal protection scheme is not available. At the end, a developer gives up and allows a certain number of copies to be illegitimate,

With Keybuster, many questions are simply solved because the software protection does not depend upon a platform, nor does it require long-term studies to get the protection running.

It can be implemented in literally any kind of software, capable to connect to the Internet and communicate over a HTTP channel. Key generation and maintenance at Keybuster allow to restrict access to applications at several levels, without the need to deliver new keys to end-users.

## Server characteristics and login

Upon login, the only essential data exchanged between Keybuster and the Internet Browser, is a session key, which has been generated by the web server. This key is guaranteed to be unique at any time during a session, but may expire after a while it is in disuse, usually after 15 minutes of inactivity. After it expired, it is necessary to repeat the login.

At no place Keybuster requires Cookies, which allows communication with almost any browser, and highly protected workstations. However, Keybuster uses a few Java Script functions and procedures, and requires your browser to enable script handling. One of the scripts used is the password hashing performed by a public domain **Message Digest 5** (MD5) routine.

Upon login, and other operations which require authentication, a password is hashed with MD5 and only this hash code is sent over the web. This way, not even an eves-dropper is able to recognize your personal password to access your account, nor may the Keybuster staff recover it from the database.

We suggest to use strong passwords, with a minimum length of 6 alphanumeric characters (A-Z, a-z, 0-9). While it is a common practice to copy the username and paste it in the password field once or more times, we discourage it, because it makes a password extremely weak.

Also it's not a good practice to use passwords which can easily be reproduced, such as your surname, company name, names of pets or children, the date of birth. The Message Digest hashing allows you to use any kind of password, and any length, without the risk of getting tampered with. While no real optimum password exist, it is still good practice to use a randomly generated password (e.g. **k24\$&Ut4e3t7\$O3Y**), or a pass phrase (e.g. **Keybuster is just what I waited for**).

Also, a password, taking only the first letter of each word of an easy pass phrase (e.g. **KijwIwf**), is pretty strong.

For no reason you should use passwords like **12345678**, **aaaaaaaa**, **abcdefg**, **asdfghj**, or with a repeating sequence of same letters for more than 2 times consecutively.

Additionally, we require to type in a always new generated sequence of random letter and number challenge, which is known at the server side only, and shown as a graphical image, just to make it less easy to robots to generate a number of passwords, trying to break in.

Further, the login trials are limited to 3 times, before any additional trial is blocked. The only way to retry to login is to reopen the browser window, starting a new session.

All these security issues together make it pretty impossible to hack an account, where, naturally, the chosen password plays the major rule.

On further releases, we may consider to add another security check: binding an account to an IP address or IP address range. This way it would be possible to login only from a working place having a specific IP.

## Key security

Upon preparation of a key for usage, a unique hash is generated for that key. This hash code represents also the public key for accessing the Keybuster key itself. While we use, for now, a combination of known data to generate the hash key, we may decide later, without any impact to existing keys, to apply a different algorithm. The data taken for the generation of hash keys remains secret, since these are our private keys for validating the public Keybuster Key.

An external application may try to generate any possible sequence, to get a list of existing keys on Keybuster. Currently, our hash keys are 128 bit wide, which ideally will take a dedicated machine, at the current state of technology, about 10.000.000 years to discover all possible combinations. In practice it would take much more time, because each validation over Internet will take some time, usually measured in seconds rather than nano-seconds, making a brute force research painfully slow.

Also, some keys may exist, but are in some way inaccessible, and the message returned will not give enough information about this particular request, but return a standard **access denied** reply.

Even though it is possible an end-user will access accidentally an enabled key, it doesn't mean the retrieved data is to any use for them, one, because Keybuster keys are application specific, with a well defined usage at our server, second, because the reply may be encrypted with a proprietary algorithm even the Keybuster team will not be able to open, having direct access to the validation data.