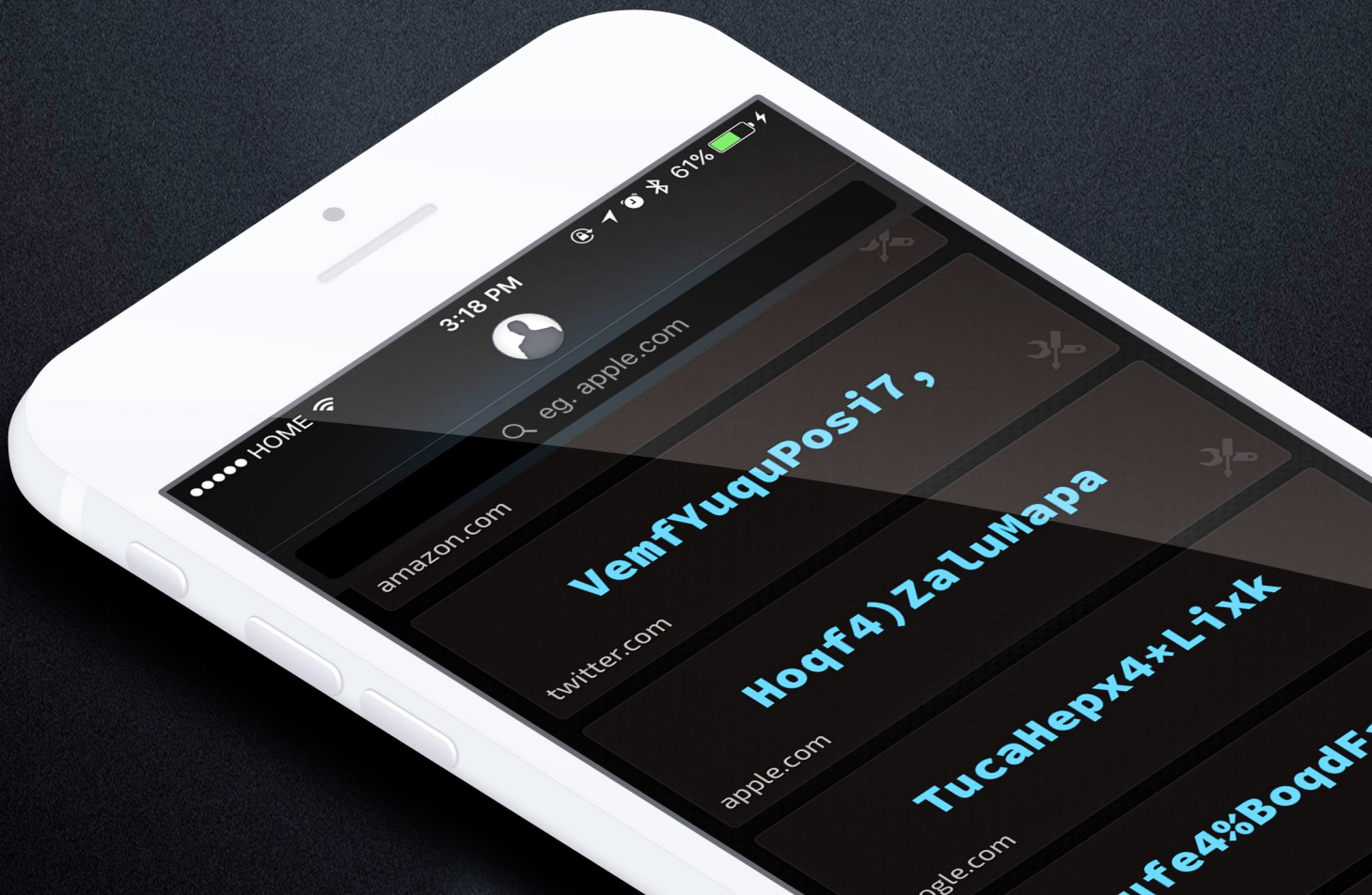
Master Password

Liberating yourself: an algorithm for freedom



THE KEY PRINCIPLE: STATELESSNESS

At the core of the Master Password solution is the principle of statelessness.

State is the technical term for “what needs to be saved”.

In most tasks, state is used to ensure you don’t need to redo your work. You write a document and save it to disk so you needn’t rewrite it every time you need it.

But not all state is reproducible. When you save your sites’ passwords on your computer, be it in a program or elsewhere, you depend on that information whenever you need to log in. If this state becomes inaccessible for any reason; you don’t have your computer handy, it breaks down, you lose it; the state is gone and there is no way to accomplish our task. We are dependent on this state. State also introduces vulnerability: private state must be protected from any and all attack vectors.

In order to free ourselves from these risks and dependencies, we need to free ourselves from state. Stateless operation implies that the computer or program can give you what you need based on nothing more than inputs you can give it.

THE MASTER PASSWORD SOLUTION

Master Password solves the password problem in a stateless manner while continuing to guarantee and to some extent even enforce good security for your sites.

Master Password implements its solution in three distinct phases:

1. Your Master Key
2. Your Site Key
3. Your Site Password

PHASE 1: YOUR IDENTITY

Your identity is defined by your master key. This key unlocks all of your doors.

Your master key is the cryptographic result of two components:

1. Your <name> (identification)
2. Your <master password> (authentication)

Your master password is your personal secret and your name scopes that secret to your identity. Together, they create a cryptographic identifier that is unique to your person.

```
masterKey       = SCRYPT[1]( key, seed, N, r, p, dkLen )

key             = <master password>
seed            = scope[2] . LEN(<name>) . <name>

N               = 32768
r               = 8
p               = 2
dkLen           = 64
```

We employ the SCRYPT cryptographic function to derive a 64-byte cryptographic key from the user's name and master password using a fixed set of parameters.

---

[1] STRONGER KEY DERIVATION VIA SEQUENTIAL MEMORY-HARD FUNCTIONS – https://www.tarsnap.com/scrypt/scrypt.pdf

[2] Table 1: Key Scopes

PHASE 2: YOUR SITE KEY

Your site key is a derivative from your master key when it is used to unlock the door to a specific site.

Your site key is the result of two components:

1. Your <site name> (identification)
2. Your <master key> (authentication)
3. Your <site counter>

Your master key ensures only your identity has access to this key and your site name scopes the key to your site. The site counter ensures you can easily create new keys for the site should a key become compromised. Together, they create a cryptographic identifier that is unique to your account at this site.

```
siteKey         = HMAC-SHA-256[1][2]( key, seed )

key             = <master key>
seed            = scope[3] . LEN(<site name>) . <site name> . <counter>
```

We employ the HMAC-SHA-256 cryptographic function to derive a 64-byte cryptographic site key from the from the site name and master key scoped to a given counter value.

[1] SHA-256, FIPS PUB 180-4 – http://dx.doi.org/10.6028/NIST.FIPS.180-4

[2] HMAC, FIPS PUB 198-1 – http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf

[3] Table 1: Master Password Key Scopes

PHASE 3: YOUR SITE PASSWORD

Your site password is an identifier derived from your site key in compliance with the site's password policy.

The purpose of this step is to render the site's cryptographic key into a format that the site's password input will accept.

Master Password declares several site password formats and uses these pre-defined password "templates" to render the site key legible.

```
template       = templates[1][ <site key>[0] % LEN( templates ) ]

for i in 0..LEN( template )
  passChars    = templateChars[2][ template[i] ]
  passWord[i]  = passChars[ <site key>[i+1] % LEN( passChars ) ]
```

We resolve a template to use for the password from the site key's first byte. As we iterate the template, we use it to translate site key bytes into password characters. The result is a site password in the form defined by the site template scoped to our site key.

This password is then used to authenticate the user for his account at this site.

[1] Table 2: Master Password Template Classes

[2] Table 3: Master Password Character Classes

KEY SCOPES

The Master Password algorithm defines several key scopes. These scopes are used to scope the key generation to a specific purpose.

Three purposes are defined:

1. **Authentication**
   The authentication scope is used when generating a key that is used for authenticating the user, such as a password.
2. **Identification**
   The identification scope is used when generating a key that is intended for the purpose of identifying the user. Identification keys are not necessarily private.
3. **Recovery**
   The recovery scope is used for generating fall-back identifiers for use in access recovery when the primary authentication mechanism has failed.

Table 1: Master Password Key Scopes

Purpose	Scope Identifier
Authentication	`com.lyndir.masterpassword`
Identification	`com.lyndir.masterpassword.login`
Recovery	`com.lyndir.masterpassword.answer`

PASSWORD TEMPLATES

In an effort to enforce increased password entropy, a common consensus has developed among account administrators that passwords should adhere to certain arbitrary password policies. These policies enforce certain rules which must be honoured for an account password to be deemed acceptable.

As a result of these enforcement practices, Master Password's site key output must necessarily adhere to these types of policies. Since password policies are governed by site administrators and not standardized, Master Password defines several password templates to make a best-effort attempt at generating site passwords that conform to these policies while also keeping its output entropy as high as possible under the constraints.

Table 2: Master Password Template Classes

Template Class	Template Set	
Maximum Security Password	anoxxxxxxxxxxxxxxxxx	axxxxxxxxxxxxxxxxxno
Long Password	CvcvnoCvcvCvcv	CvcvCvcvCvccno
	CvcvCvcvnoCvcv	CvccnoCvccCvcv
	CvcvCvcvCvcvno	CvccCvccnoCvcv
	CvccnoCvcvCvcv	CvccCvccCvcvno
	CvccCvcvnoCvcv	CvcvnoCvccCvcc
	CvccCvcvCvcvno	CvcvCvccnoCvcc
	CvcvnoCvccCvcv	CvcvCvccCvccno
	CvcvCvccnoCvcv	CvccnoCvcvCvcc
	CvcvCvccCvcvno	CvccCvcvnoCvcc
	CvcvnoCvcvCvcc	CvccCvcvCvccno
	CvcvCvcvnoCvcc	
Medium Password	CvcnoCvc	CvcCvcno
Short Password	Cvcn	
Basic Password	aaanaaan	aannaaan
	aaannaaa	
PIN	nnnn	

A Master Password template is a string of characters, where each character identifies a certain character class. As such, the template specifies that the output password should be formed by substituting each of the template's character class characters by a chosen character from that character class.

Table 2: Master Password Character Classes

Character Class	Character Set
V (vowels)	`AEIOU`
C (consonants)	`BCDFGHJKLMNPQRSTVWXYZ`
v (vowels)	`aeiou`
c (consonants)	`bcdfghjklmnpqrstvwxyz`
A (alphabetic)	`AEIOUBCDFGHJKLMNPQRSTVWXYZ`
a (alphabetic)	`AEIOUaeiouBCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz`
n (numeric)	`123456789`
o (other)	`@&%?,=[]_:-+*$#!'^~;()/.`
X (union set)	`AEIOUaeiouBCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz0123456789!@#$%^&*()`