

UNIVERSITY NAME

DOCTORAL THESIS

---

# Hierarchical deterministic wallet

---

*Author:*

Daniele FORNARO

*Supervisor:*

Daniele MARAZZINA

*A thesis submitted in fulfillment of the requirements  
for the degree of Mathematical Engineering*

*in the*

Research Group Name  
Department or School Name

January 19, 2018



## Declaration of Authorship

I, Daniele FORNARO, declare that this thesis titled, “Hierarchical deterministic wallet” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

---

Date:

---



*“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”*

Dave Barry



UNIVERSITY NAME

# *Abstract*

Faculty Name  
Department or School Name

Mathematical Engineering

**Hierarchical deterministic wallet**

by Daniele FORNARO

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...





## *Acknowledgements*

The acknowledgments and the people to thank go here, don't forget to include your project advisor...



# Contents

<b>Declaration of Authorship</b>	<b>iii</b>
<b>Abstract</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>ix</b>
<b>1 Elliptic Curve Geometry</b>	<b>1</b>
1.1 Introduction	1
1.2 Elliptic Curve over $\mathbb{F}_p$	1
1.2.1 Operations	1
Symmetry	1
Point addition	2
Scalar multiplication	3
1.2.2 Group order	3
Cyclic subgroups	3
1.3 Bitcoin private-public key cryptography	4
1.3.1 Bitcoin Elliptic Curve	4
1.3.2 Bitcoin keys representation and addresses	4
Uncompressed public key	4
Compressed public key	5
WIF Private Key	5
Address	5
<b>2 Wallet</b>	<b>7</b>
2.1 Nondeterministic ( <i>random</i> ) Wallet	7
2.1.1 Pros and Cons	7
2.2 Deterministic Wallets	8
2.2.1 Deterministic Wallet <i>type-1</i>	8
2.2.2 Pros and Cons	9
2.3 Hierarchical deterministic Wallets	9
2.3.1 Key Concept	9
Seed	9
Extended Key	9
Mnemonic and Passphrase	9
2.3.2 Pros and Cons	9
<b>3 Seed to Master Private Key</b>	<b>11</b>
3.1 Functional explanation	11
3.2 Code implementation	11

<b>4</b>	<b>Child Key Derivation</b>	<b>13</b>
4.1	Functional explanation . . . . .	13
4.2	Normal derivation . . . . .	13
4.2.1	Derive public child from public parent . . . . .	13
4.2.2	Possible Risk . . . . .	13
4.3	Hardened derivation . . . . .	13
<b>5</b>	<b>Mnemonic to Seed</b>	<b>15</b>
5.1	Functional explanation . . . . .	15
5.2	BIP 39 derivation . . . . .	15
5.2.1	Mnemonic generation . . . . .	15
5.2.2	Seed derivation . . . . .	15
5.3	Electrum derivation . . . . .	15
5.3.1	Mnemonic generation . . . . .	15
5.3.2	Seed derivation . . . . .	15
5.4	BIP39 vs Electrum derivation . . . . .	15
<b>6</b>	<b>How to use a HD Wallet</b>	<b>17</b>
6.1	Multi-coin wallet BIP 44 . . . . .	17
<b>A</b>	<b>Frequently Asked Questions</b>	<b>19</b>
A.1	How do I change the colors of links? . . . . .	19
	<b>Bibliography</b>	<b>21</b>

# List of Figures

1.1	Points on the Elliptic Curve $y^2 = x^3 - 7x + 10 \pmod{p}$ , with $p =$ 19, 97, 127, 487 . . . . .	2
1.2	Elliptic Curve $y^2 = x^3 - 7x + 10 \pmod{97}$ . . . . .	2



# List of Tables





# List of Abbreviations

**LAH** List Abbreviations **Here**  
**WSF** What (it) Stands For



# Physical Constants

Speed of Light  $c_0 = 2.997\,924\,58 \times 10^8 \text{ m s}^{-1}$  (exact)



# List of Symbols

$a$	distance	m
$P$	power	W (J s <sup>-1</sup> )
$\omega$	angular frequency	rad



*For/Dedicated to/To my...*





## Chapter 1

# Elliptic Curve Geometry

### 1.1 Introduction

Bitcoin security is based on public and private key cryptography. The main concept is that it is simple to compute the public key, knowing the private, but it is infeasible to calculate the private key, knowing the public.

In order to obtain this result a particular Elliptic Curve is used.

### 1.2 Elliptic Curve over $\mathbb{F}_p$

A point  $Q$ , which coordinates are  $x$  and  $y$ , belong to an Elliptic Curve if and only if  $Q$  satisfies the following equation:

$$y^2 = x^3 + ax + b \quad \text{over } \mathbb{F}_p \quad (1.1)$$

Where  $\mathbb{F}_p$  is the finite field defined over the set of integers modulo  $p$  and  $a$  and  $b$  are the coefficients of the curve.

We can rewrite the equation 1.1 in the following way:

$$y^2 = x^3 + ax + b \quad \text{mod } p \quad (1.2)$$

Figure 1.1 shows some examples of Elliptic Curve over  $\mathbb{F}_p$  with  $a = -7$  and  $b = 10$

#### 1.2.1 Operations

A point on the Elliptic Curve has some particular properties:

- Symmetry
- Point addition
- Scalar multiplication

##### Symmetry

For every point in the  $x$  axis exists two points in the  $y$  axis. Suppose that a point  $P(x, y)$  belongs to the Elliptic Curve, then it must satisfy the equation 1.1. So it is easy to prove that the point  $Q(x, p - y)$  belongs to the curve too.

Furthermore we have  $P = -Q$ , from the moment that  $P + Q = 0$  (see addition below).



FIGURE 1.1: Points on the Elliptic Curve  $y^2 = x^3 - 7x + 10 \pmod{p}$ , with  $p = 19, 97, 127, 487$

### Point addition

We need to change our definition of addition in order to make it works in  $\mathbb{F}_p$ . In this framework we claim that if three points are aligned over the finite field  $\mathbb{F}_p$ , then they have zero sum.

So  $P + Q = R$  if and only if  $P, Q$  and  $-R$  are aligned, in the sense shown in figure 1.2



FIGURE 1.2: Elliptic Curve  $y^2 = x^3 - 7x + 10 \pmod{97}$

The equations for calculating point additions are the follow:  
Suppose that  $A$  and  $B$  belong to the Elliptic Curve.

$$A = (x_1, y_1) \quad B = (x_2, y_2)$$

Let's defined  $A + B := (x_3, y_3)$

So we have:

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } x_1 = x_2 \end{cases}$$

$$\begin{aligned} x_3 &= s^2 - x_1 - x_2 \pmod{p} \\ y_3 &= s(x_1 - x_3) - y_1 \pmod{p} \end{aligned}$$

### Scalar multiplication

Once defined the addition, any multiplication can be defined as:

$$nP = \underbrace{P + P + \dots + P}_{n \text{ times}}$$

When  $n$  is a very large number can be difficult or even infeasible to compute  $nP$  in this way, but we can use the *double and add algorithm* in order to perform multiplication in  $\mathcal{O}(\log n)$  steps.

### 1.2.2 Group order

An elliptic curve defined over a finite field is a group and so it has a finite number of points. This number is called order of the group.

If the prime order is a very large number, it is impossible to count all the point in that field, but there is an algorithm that allows to calculate the order of a group in a fast and efficient way: *Schoof's algorithm*.

### Cyclic subgroups

Let's consider a generic point  $P$ , we have:

$$nP + mP = \underbrace{P + \dots + P}_{n \text{ times}} + \underbrace{P + \dots + P}_{m \text{ times}} = \underbrace{P + \dots + P}_{n+m \text{ times}} = (n+m)P$$

So multiple of  $P$  are closed under addition and this is enough to prove that the set of the multiples of  $P$  is a cyclic subgroup of the group formed by the elliptic curve.

The point  $P$  is called **generator** of the cyclic subgroup.

**Remark** The order of  $P$  is linked to the order of the elliptic curve by Lagrange's theorem, which states that the order of a subgroup is a divisor of the order of the parent group.

**Remark** If the order of the group is a prime number, all the point  $P$  generate a subgroup with the same order of the group.

### 1.3 Bitcoin private-public key cryptography

#### 1.3.1 Bitcoin Elliptic Curve

Bitcoin uses a specific Elliptic Curve defined over the finite field of the natural numbers, where  $a = 0$  and  $b = 7$ .

The equation 1.1 becomes:

$$y^2 = x^3 + 7 \pmod{p} \quad (1.3)$$

The  $\pmod{p}$  (modulo prime number) indicates that this curve is over a finite field of prime order  $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ .

The *order* of this Elliptic Curve is a very large prime number, close to  $2^{256}$ , but smaller than  $p$ .

Let's consider a particular point  $G$ , called generator, with:

$$\begin{aligned} x = & 79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798 \\ y = & 483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8 \end{aligned}$$

From the moment that the order of the group is a prime number, the order of any subgroup is equal to the order of the entire group. In particular the order of the subgroup generated by  $G$  is equal to *order*.

**Definition** A private key is a number chosen in the range between 1 and *order*.

**Definition** A public key  $W$  is a point in the Bitcoin EC, derived from a private key  $k$  in the following way:

$$W = k \cdot G \quad (1.4)$$

Where the multiplication between  $k$  and  $G$  is defined in the previous chapter.

This is a *one way* function, in the sense that computing the scalar multiplication, knowing the private key is simple, but make the reverse is infeasible.

**Remark** It is infeasible to calculate a private key knowing the public key.

#### 1.3.2 Bitcoin keys representation and addresses

In order to make it easy to store and recognise keys, some encodings were designed.

A public key, a point in the EC, can be represented in two ways: *uncompressed* or *compressed*.

##### Uncompressed public key

An uncompressed public key is represented in hexadecimal digits, and it is obtained simply concatenating the  $x$  coordinate with the  $y$  coordinate and adding 04 at the beginning, for a total of 130 hexadecimal digits.

Example of an uncompressed public key:

0450863AD64A87AE8A2FE83C1AF1A8403CB53F53E486D8511DAD8A04887E5B235  
22CD470243453A299FA9E77237716103ABC11A1DF38855ED6F2EE187E9C582BA6

### Compressed public key

A compressed public key is obtained simply taking the  $x$  coordinate and adding 02 at the beginning if the  $y$  coordinate is even, 03 otherwise.

This is due to the *symmetry properties* of a point of the EC.

Example of a public key compressed:

0250863AD64A87AE8A2FE83C1AF1A8403CB53F53E486D8511DAD8A04887E5B2352

### WIF Private Key

WIF stands for wallet import format and is the standard way used to write down a private key.

- Add a version number (80 for Bitcoin) in front of the private key, in order to recognize quickly for what cryptocurrency that private key was used.
- Add 01 at the end of the private key if you want a WIF *compressed*, none if you want a WIF *uncompressed*. The difference between these two types is that from a *compressed* private key a *compressed* public key is expected and from a *uncompressed* private key a *uncompressed* public key is expected.
- Add a checksum at the end, obtained applying the SHA256 function twice to the string previously obtained, take the first 4 bytes (8 hexadecimal digits) and put them at the end of the string.
- Compute the Base58Encode, obtaining a 52 digit string.

Example of private key WIF:

KwdMAjGmerYanjeui5SHS7JkmpZvVipYvB2LJGU1ZxJwYvP98617

### Address

Among the Bitcoin transactions, one of the most used is a *Pay-to-PubkeyHash*, meaning that in the transaction you will not write directly the public key, but the hash of that public key.

The hash function used in this framework is the HASH160 function, applied to the *compressed* public key. This is an irreversible procedure, so you cannot obtain the public key from the public key hash.

In order to obtain a valid Bitcoin address, it is needed to encode the *PubkeyHash* in base58, adding first the version in front, the checksum at the end and then encode everything with Base58Encode, obtaining a 34 digit string.

Example of an Address:

1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2



## Chapter 2

# Wallet

A Bitcoin wallet is a structure used to store keys.

There are different type of wallet:

- Nondeterministic (*random*) Wallet
- Deterministic Wallet

**Remark** *Bitcoin wallets contains keys, not coins. Coins are in the Blockchain.*

### 2.1 Nondeterministic (*random*) Wallet

A nondeterministic wallet is the simplest type of wallet. Each Key is randomly and independently generated.

- (i) Consider a *Discrete Uniform Random Variable*

$$X \sim \mathcal{U}(S)$$

Where  $S$  is the finite set of natural number in the range from 1 to *order*.

- (ii) Take some realizations  $k_1, k_2 \dots k_n$  of  $X$  using enough entropy to make these numbers (*private keys*) impossible to guess.

$$k_1 = X(\omega_1) \quad k_2 = X(\omega_2) \quad \dots \quad k_n = X(\omega_n)$$

- (iii) Go back to point (i) every time new *private keys* are needed.

#### 2.1.1 Pros and Cons

Let's focus on the good and bad aspect of this wallet.

<i>Random Wallet</i>	
Pros	Cons
<ul style="list-style-type: none"> <li>• Easy to implement</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to find <u>real</u> new entropy for every new <i>private key</i>.</li> <li>• Every time new <i>private keys</i> are needed, you need to make new back up.</li> <li>• Difficult to store or back up in a <i>non digital way</i>. Awkward to write it down all yours keys on a paper.</li> </ul>

The use of *random wallet* is strongly discouraged for anything other than simple test. There are no good reason to use it.

## 2.2 Deterministic Wallets

A deterministic wallet is a more sophisticated one, in which every key is generated from a common "*seed*". This means that knowing the *seed* means also to know all the keys in the wallet.

There are different types of deterministic wallets, in this text we will analyze three main types:

- Deterministic Wallet *type 1*
- Deterministic Wallet *type 2*
- Hierarchical Deterministic Wallet

These wallet are in increasing order of complexity.

### 2.2.1 Deterministic Wallet *type-1*

The Deterministic Wallet *type-1* is one of the simplest Wallet among the deterministic ones. Each key is generated adding a number in a sequential order to the *seed* and then computing an *hash* function such as the **SHA256** function.

Let's see how it works:

- (i) Generate a *seed*, a random number from a *Discrete Uniform Random Variable*

$$seed = X(\omega) \quad X \sim \mathcal{U}(S)$$

Where  $S$  is the finite set of natural number in the range from 1 to *order*.

- (ii)



### **2.2.2 Pros and Cons**

## **2.3 Hierarchical deterministic Wallets**

### **2.3.1 Key Concept**

Seed

Extended Key

Mnemonic and Passphrase

### **2.3.2 Pros and Cons**



## **Chapter 3**

# **Seed to Master Private Key**

### **3.1 Functional explanation**

### **3.2 Code implementation**



## **Chapter 4**

# **Child Key Derivation**

### **4.1 Functional explanation**

### **4.2 Normal derivation**

#### **4.2.1 Derive public child from public parent**

#### **4.2.2 Possible Risk**

### **4.3 Hardened derivation**



## **Chapter 5**

# **Mnemonic to Seed**

### **5.1 Functional explanation**

### **5.2 BIP 39 derivation**

#### **5.2.1 Mnemonic generation**

#### **5.2.2 Seed derivation**

### **5.3 Electrum derivation**

#### **5.3.1 Mnemonic generation**

#### **5.3.2 Seed derivation**

### **5.4 BIP39 vs Electrum derivation**





## **Chapter 6**

# **How to use a HD Wallet**

### **6.1 Multi-coin wallet BIP 44**



## Appendix A

# Frequently Asked Questions

### A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

```
\hypersetup{urlcolor=red}, or  
\hypersetup{citecolor=green}, or  
\hypersetup{allcolor=blue}.
```

If you want to completely hide the links, you can use:

```
\hypersetup{allcolors=.}, or even better:  
\hypersetup{hidelinks}.
```

If you want to have obvious links in the PDF but not the printed text, use:

```
\hypersetup{colorlinks=false}.
```



# Bibliography

- Arnold, A. S. et al. (Mar. 1998). "A Simple Extended-Cavity Diode Laser". In: *Review of Scientific Instruments* 69.3, pp. 1236–1239. URL: <http://link.aip.org/link/?RSI/69/1236/1>.
- Hawthorn, C. J., K. P. Weber, and R. E. Scholten (Dec. 2001). "Littrow Configuration Tunable External Cavity Diode Laser with Fixed Direction Output Beam". In: *Review of Scientific Instruments* 72.12, pp. 4477–4479. URL: <http://link.aip.org/link/?RSI/72/4477/1>.
- Wieman, Carl E. and Leo Hollberg (Jan. 1991). "Using Diode Lasers for Atomic Physics". In: *Review of Scientific Instruments* 62.1, pp. 1–20. URL: <http://link.aip.org/link/?RSI/62/1/1>.