■ Bitcoin Explorerのダウンロード

本書では、Bitcoin Explorer 3.2.0を利用します。以降では、次のサイトよりダウンロードする方法を説明します。

・Bitcoin Explorerのダウンロードサイト

URL https://github.com/libbitcoin/libbitcoin-explorer/wiki/Download-BX

■ Bitcoin Explorerのインストール

まず、インストールディレクトリを作成して移動します(**コマンド2-2-1**)。コマンドの後ろにセミコロン(**;**)を入力し、続いて次のコマンドを入力すると、同時に複数のコマンドを実行できます。本書ではこの方法を多用するため、覚えてください。

本書に記載のコマンドはすべて本書サポートページよりダウンロード可能です。詳細はvページ(「掲載コマンド等のダウンロード」)を参照してください。インストール作業などは間違えると後々問題となりやすいため、コピーして利用してください。ただし、本書では手を動かすことに重きを置いているため、インストール作業以外のコマンドは自身で入力することを推奨します。どうしてもエラーが出る場合などに、タイプミスかどうかを切り分ける用途で利用してください。

▽コマンド2-2-1　インストールディレクトリの作成・移動

```
$ mkdir -p ~/bx ; cd ~/bx ↵
```

続いて、**wget**コマンドで実行形式ファイルをダウンロードします(**コマンド2-2-2**)。

▽コマンド2-2-2　wgetでモジュールをダウンロード

```
$ wget https://github.com/libbitcoin/libbitcoin-explorer/releases/download/v3.2.0/bx-linux-
x64-qrcode ↵
```

実行形式ファイル(**bx-linux-x64-qrcode**)の名前が長いので、**bx**に変更します。併せて、実行権限を付与します。権限が「**rwx**」になっていればOKです(**コマンド2-2-3**)。

▽コマンド2-2-3　ファイル名を変更し、実行権限を付与

```
$ mv bx-linux-x64-qrcode bx ; chmod 700 bx ; ls -l ↵
total 4924
-rwx------ 1 bc01 bc01 5037768  5月 26  2017 bx
```

bxを環境変数**PATH**に追加します。**vi**エディタで「**~/.bashrc**」を開き、末尾に次の行を追記してください(**コマンド2-2-4**)。


```
PATH="${PATH}:${HOME}/bx"
```

▽コマンド2-2-4　PATHの追加(viエディタを起動)

```
$ vi ~/.bashrc ↵
```

sourceコマンドで読み込んで変更を有効にしましょう(**コマンド2-2-5**)。

▽コマンド2-2-5　.bashrcを読み込む

```
$ source ~/.bashrc ↵
```

whichコマンドに**bx**を指定して実行したとき、**bx**のパスが表示されるとOKです(**コマンド2-2-6**)。

▽コマンド2-2-6　パスが通ったか確認

```
$ which bx ↵
/home/bc01/bx/bx
```

Bitcoin Explorerでは多くのコマンドが提供されています。次のサイトでは、提供されているコマンドとその使い方を確認できます。

・「Home：libbitcoin/libbitcoin-explorer Wiki(GitHub)」

URL https://github.com/libbitcoin/libbitcoin-explorer/wiki

2.3：Bitcoin Coreのインストール

Bitcoin Coreは、ビットコインネットワークの公式のクライアントソフトです。本書では後半から利用しますが、ビルドとブロックのダウンロードに時間がかかるため、本書を読み進めながら並行してインストールすることをお勧めします。

本書では、Bitcoin Core 0.16.0を利用します。

■ 必要なパッケージのインストール

bc01ユーザでログインし、必要なパッケージをインストールします(**コマンド2-3-1**)。

Chapter4

鍵、アドレス、ウォレット

仮想通貨には、実物のコインや紙幣は存在しません。それでは、どのように所有権を特定したり、送金したりしているのでしょうか。本章では「鍵」「アドレス」「ウォレット」の役割について説明します。

4.1：所有権を特定する「鍵」と「錠」

実物のコインや紙幣がないビットコインでは、買い物をする際にどのように所有者が代わるのでしょうか。

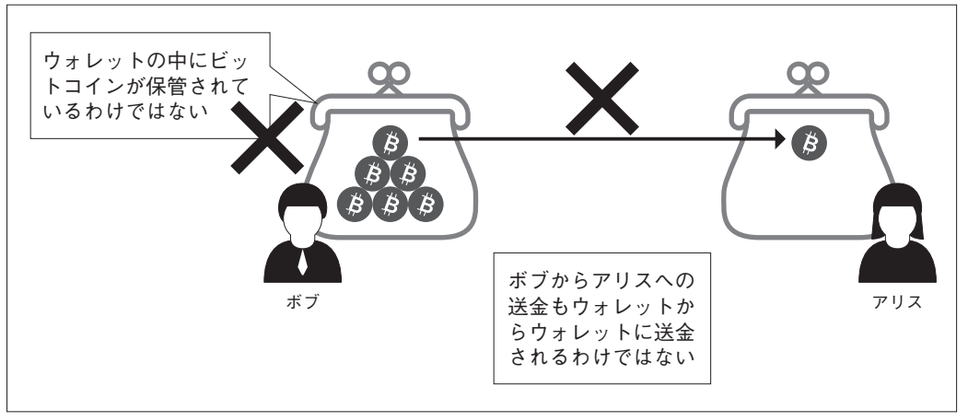
Aさんが所有するビットコインを別の人に移す場合は、Aさん自身の「鍵」で「錠」を解除して、新たな所有者の「鍵」でしか解除できない「錠」をかけておくイメージです。分散台帳にも所有権の移動が表現されます。

鍵と錠とは、具体的には楕円曲線暗号によって生成された秘密鍵（鍵）と公開鍵（錠）です（錠のかけ方や解除方法は次章で説明）。秘密鍵は参加者によってランダムに選ばれる数字ですが、たいていの場合は専用のソフトウェアによって自動生成されます。秘密鍵のみがビットコインの所有権を証明する唯一の手段なので、絶対に他人に渡したり、なくしたりしてはいけません。万が一、紛失した場合はビットコインを永久に利用できなくなります。また、秘密鍵が漏洩して、悪意のある第三者が所有権を移した場合も失われます。

4.2：鍵を管理する「ウォレット」

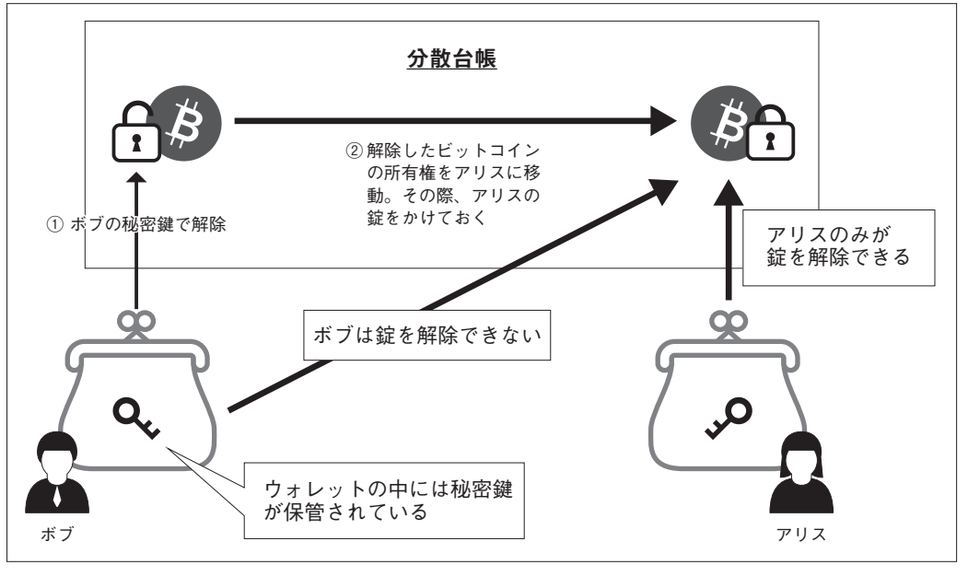
「ウォレット」は、狭義では鍵を管理するためのものです。とりわけ秘密鍵を管理するものですが、公開鍵のみを管理する場合もあります。送金、残高の確認、鍵やアドレスの管理といった機能を提供するユーザインターフェースを持つアプリケーションを指すこともあります。ウォレット、つまりお財布というイメージですが、ウォレットの中にビットコインが保管されているわけではありません（**図4-2-1**）。

▽図4-2-1　ウォレットの誤ったイメージ



ビットコインは分散台帳内で管理されています。所有権の移動を鍵と錠で例えましたが、送金時のウォレットの利用イメージは**図4-2-2**のようになります。

▽図4-2-2　送金時のウォレットの利用イメージ（ボブからアリスへの送金の場合）



①分散台帳上でボブに所有権があるビットコインに錠がかけられている。所有権をアリスへ移す場合は、ボブの秘密鍵で解除する

②所有権を移動する際に、アリスの秘密鍵のみが解除できる錠をかける

Chapter10 ウォレット

Bitcoin Coreを使ってウォレットを操作してみましょう。

10.1：ウォレットを操作するための準備

まずは事前準備として、Bitcoin Coreを停止した状態で**コマンド10-1-1**の作業を行ってください。

testnetの場合、デフォルトで**~/.bitcoin/testnet3**直下のウォレットファイル**wallet.dat**をロードしますが、同ディレクトリに**wallets**ディレクトリがあれば、その直下の**wallet.dat**をロードするようになります。ディレクトリで用途を分けたほうがわかりやすいので、本書では**wallets**ディレクトリを作成します。なお、すでに存在する可能性も考慮して、まずはクリーンアップ(**rm**コマンド)から実施しています(**コマンド10-1-1**)。

▽コマンド10-1-1　walletsディレクトリを作成

```
$ rm -fr ~/.bitcoin/testnet3/wallets ↵
$ mkdir -p ~/.bitcoin/testnet3/wallets ↵
$ chmod 700 ~/.bitcoin/testnet3/wallets ↵
```

bitcoindを起動する前に、作業用に**~/work/wallet**ディレクトリを作成して移動してください(**コマンド10-1-2**)。これは、後ほどウォレットファイルのバックアップを取る際に、起動時のディレクトリにバックアップされるためです。ウォレットファイルのバックアップはこのディレクトリに保存することにします。

▽コマンド10-1-2　作業用ディレクトリを作成

```
$ mkdir -p ~/work/wallet ; cd ~/work/wallet ↵
```

Bitcoin Coreを起動します(**コマンド10-1-3**)。**bcit help**を実行して正常に結果が返ってくるまで待ってください。「**error code: -28**」が返ってきた場合は起動中です。

▽コマンド10-1-3　Bitcoin Coreを起動

```
$ bitcoind -daemon ↵
Bitcoin server starting
```

起動すると、**wallets**ディレクトリに**wallet.dat**が作成されていることが確認できます(**コマンド10-1-4**)。

▽コマンド10-1-4　wallet.datを確認

```
$ ls -ltr ~/.bitcoin/testnet3/wallets ↵
total 1360
-rw------- 1 bc01 bc01       0  4月  5 23:32 db.log
drwx------ 2 bc01 bc01    4096  4月  5 23:33 database
-rw------- 1 bc01 bc01 1388544  4月  5 23:33 wallet.dat
```

10.2：ウォレットファイルのダンプとバックアップ

ウォレットファイルのダンプとバックアップの取得方法を説明します。

■ウォレットファイルのダンプを出力する

「ダンプ」とは、ウォレットファイルの中身をテキストに書き出したものです。

bcit dumpwalletにファイル名を指定すると、そのファイル名でダンプファイルが生成され、ファイルのパスが出力されます(**コマンド10-2-1**)。

▽コマンド10-2-1　ウォレットのダンプを出力

```
$ bcit dumpwallet dumpwallet.dat.01 ↵
{
  "filename": "/home/bc01/work/wallet/dumpwallet.dat.01"
}
```

■ウォレットファイルのバックアップを取る

バックアップは、**wallet.dat**ファイルのバックアップです。**コマンド10-2-2**のように実行すると安全にバックアップファイルが取得できます。

bcit backupwalletにファイル名を指定すると、そのファイル名でバックアップファイルを生成します。**bcit dumpwallet**と違い、出力先は表示されませんが、バックアップファイルは作業ディレクトリに出力されます。

Chapter17
ホットウォレットのセキュリティ

本章ではホットウォレットを採用する場合のセキュリティについて、主にシステムアーキテクチャや運用設計の観点で説明します。

17.1：秘密鍵をホットウォレットとするか検討する

はじめに検討すべきことは、本当にホットウォレットを採用すべきかどうかです。ホットウォレットは、インターネットに接続したオンライン環境下に秘密鍵があるため、サイバー攻撃のリスクが高くなります。秘密鍵は送金のために必要なので、送金を必要としない、またはめったに必要としないのであれば、ホットウォレットではなくコールドウォレットを採用すべきです。

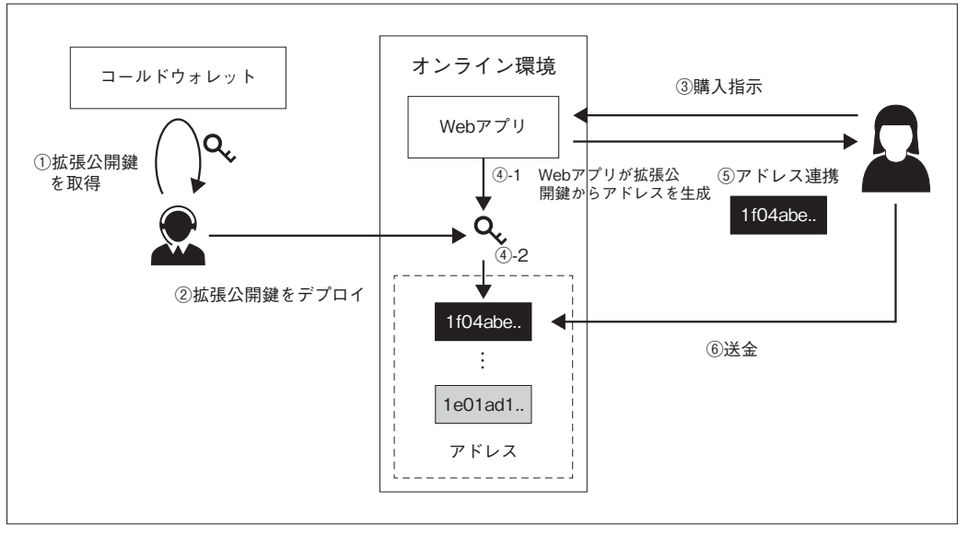
例えば、ECサイトを考えてみましょう。考えられるサービスとして、ビットコインによる支払いに対応する場合です。キャンセルなどの特殊な場合を除いて、基本的にはユーザからビットコインを受け取るだけであるため、そもそも秘密鍵を利用するケースは通常業務では発生しないはずです。返金などの際は、コールドウォレットから返却するという運用で、運用負荷的に問題ないのであれば、ホットウォレットを採用すべきではありません。

一方でユーザからの送金のためにアドレスを生成する必要があります。送金先アドレスを購入のたびに生成してECサイト上に表示するような場合、コールドウォレット側でそのつどアドレスを生成してECサイト側に連携するのは、運用負荷の観点で現実的ではありません。

このような場合は、拡張公開鍵を利用します。コールドウォレット側でHDウォレットを生成し、拡張公開鍵をECサイトのサーバに格納して、拡張公開鍵からアドレスを生成する設計にするとよいでしょう。

アーキテクチャと運用のイメージは、**図17-1-1**のとおりです。

▽図17-1-1　拡張公開鍵を利用する設計例



①コールドウォレットで拡張公開鍵を生成して取得する
②拡張公開鍵を公開サーバにデプロイする
――以降は、ユーザからの購入指示のたびに発生する
③ユーザが購入指示をWebアプリに出す
④Webアプリが拡張公開鍵を使ってアドレスを生成する
⑤生成したアドレスをユーザに連携する（ECサイトに表示する）
⑥ユーザがアドレスにビットコインを送金する

このように運用することで、秘密鍵をオンライン環境下に格納する必要がなくなります。キャンセルなどの際は、コールドウォレットを操作してユーザが提示したアドレスに送金すればよいだけです。

17.2：拡張公開鍵の利用時の注意点

それでも返金などのために、一部の秘密鍵はホットウォレットとして利用したい場合も考えられます。

しかし、インターネットに直接接続されている公開サーバのように攻撃を受けやすい場所に秘密鍵を保管するのはリスクが高いため、秘密鍵は攻撃を受けにくい、より内側のサーバに保管しておくのがよいでしょう。アドレス生成のための拡張公開鍵と、送金用の拡張秘密鍵を格納するサーバをセキュリティレベルで分ける考え方です。