2017 年全国大学生信息安全竞赛

作品报告

作品名称：一种公平的异步合同签署系统

电子邮箱：hantianxu1996@163.com

提交日期：2017-05-29

填写说明

1. 所有参赛项目必须为一个基本完整的设计。作品报告书旨在能够清晰准确地阐述（或图示）该参赛队的参赛项目（或方案）。

2. 作品报告采用A4纸撰写。除标题外，所有内容必需为宋体、小四号字、1.5倍行距。

3. 作品报告中各项目说明文字部分仅供参考，作品报告书撰写完毕后，请删除所有说明文字。(本页不删除)

4. 作品报告模板里已经列的内容仅供参考，作者可以在此基础上增加内容或对文档结构进行微调。

5. 为保证网评的公平、公正，作品报告中应避免出现作者所在学校、院系和指导教师等泄露身份的信息。

目录

摘要

现如今，电子商务作为新兴产业，已经在人们的生产与生活中占据了重要地位。电子商务的买家与卖家通过网络进行交易，大多是基于公平交换协议。在目前大多数的公平交换协议中，一个可信的第三方在其中发挥着关键作用。然而，第三方的加入使得通信方对其产生了极强的依赖性，一方面降低了通信双方的通信效率，加大了通信双方在时间和金钱上的消耗；另一方面，第三方的强大作用使其称为被黑客攻击的重点对象，一旦其被攻破，对通信双方造成的损失是不可挽回的。因此，能够提供一个去中心化的公平交换系统，是通信方迫切需求的。

本文设计并实现了名为“FairContract”的 PC 客户端——一种公平的异步合同签署系统。两个可能互不信任的用户，可以借助 FairContract 完成对合同签名的公平交换过程。该系统具有以下特点：

1. 去中心化。签名交换过程完全由双方控制，具有很强的安全性
2. 公平性。本系统采用罚金进制，但凡有一方失信，这部分罚金会支付给另一方作为惩罚；
3. 异步签署。即双方不必同时交出自己的签名，这给双方提供了一定的自由性和灵活性，便于平衡双方在实际交易中的时间冲突。
4. 签名与交易等价结合。即将合同文本以及代表个人身份的信息等价转换为一个比特币账户的公私钥，间接地将相关交易的生效等价为相关对合同的签名生效，这为比特币系统和数字签名的联系架起了一座桥梁。

在设计与实现方面，本系统完成了三个基本的功能模块：合同与比特币公私钥的等价转换、对等价转换的安全验证计算、以及签名的交换过程。其中，安全验证计算用到了基于椭圆曲线密码的零知识证明方法，保证了交换双方提供合同的正确性和一致性；而签名的交换过程运用隔离见证技术，通过将交易与见证隔离，实现了对双方各自合同签名的保护作用。

本系统将比特币与传统公平交易系统进行充分结合，并引入了区块链、隔离见证、零知识证明等密码学工具，完整的设计并实现了基于比特币交易网络的具有去中心化、公平性、异步性和签名交易等价性的合同签署方案系统。通过本系统的实现，我们相信，传统的方案与新兴的比特币分布式可信网络的充分结合将会是未来的发展趋势。

关键词：合同签署，公平，异步，比特币网络

第一章 作品概述

1.1 背景分析

1.1.1 公平交换与第三方

公平交换[1]由于其广泛的应用场景和固有的特点已经被密码学界研究了 30 多年。一直以来，无论是逐比特交换协议还是 TTP 公平交换协议(在线或离线)都没有在保证公平性和摒弃第三方依赖上实现完全的统一。随着中心攻击问题的愈加严重，在线第三方逐渐淡出历史舞台，取而代之的是以离线第三方为仲裁者的仲裁协议。

趋势表明，层出不穷的安全攻击行为使得人们对去第三方的公平交换协议的需求越来越迫切。然而，完全去除第三方就意味着更高复杂度的算法和更加复杂的协议，在这一点上如何实现突破，也是密码学家在深入研究的课题。

1.1.2 以比特币系统为代表的去中心化潮流

2009 年，中本聪[2]发表比特币创世论文《比特币：一种点对点的电子现金系统》。点对点便意味着对中心化的彻底抛弃，比特币也成为了唯一一个典型的、并且安全性极高的去中心化系统。

由于比特币系统对脚本编程的开放，使得其底层技术区块链应用的开发一直在如火如荼地进行着。比特币系统的源代码公布后，其极强的安全性已经不言而喻，区块链也因此成为去第三方应用的必选平台。

1.1.3 区块链技术

区块链（Blockchain）是由节点参与的分布式数据库系统，也可以将其理解为账簿系统，它的特点是不可更改，不可伪造。它是比特币的一个重要概念，包含了完整比特币区块链的副本，记录了其代币的每一笔交易。通过这些信息，我们可以找到每一个地址，在历史上任何一点所拥有的价值。

区块链是由一串使用密码学方法产生的数据块按时序所组成的数据链。每一个区块

都包含了上一个区块的哈希值（hash），从创始区块开始连接到当前区块，形成块链。每一个区块都确保按照时间顺序在上一个区块之后产生，否则前一个区块的哈希值是未知的。这些特征使得比特币的双花非常困难。区块链是保证比特币系统安全性的核心创新。

区块链体系结构的核心优势在于：首先，任何节点都可以创建交易，在经过一段时间的确认之后，就可以合理地确认该交易是否为有效，对于试图重写或者修改交易记录而言，它的成本是非常高的。因此，区块链可有效地防止双花问题的发生。其次，区块链实现了两种记录：交易以及区块。交易是被存储在区块链上的实际数据，而区块则是记录确认某些交易是在何时，以及以何种顺序成为区块链数据库的一部分。交易是由参与者在正常过程中使用系统所创建的（在加密数字货币的例子中，一笔交易是由 Bob 将代币发送给 Alice 所创建的），而区块则是由我们称之为矿工的单位负责创建。这两种记录的叠加保证了数据的完整性和可追溯性。

1.2 场景构建

我们开发的系统基于以下场景：

Alice 和 Bob 各自有一份合同需要对方签署，需要交换各自对于对方合同的签名（这两份合同可以相同）。他们存在以下顾虑：

1. **Alice 不信任 Bob**，她不愿意先交出自己的签名，担心 Bob 拿到她的签名后不会将自己的签名交出。这样以来，如果没有第三方的存在，就无法对 Bob 的失信行为进行界定，从而使 Alice 蒙受损失。

   同样的，**Bob 也不信任 Alice**，因为 Alice 也可以通过上述的方式让先交出签名的 Bob 蒙受损失。

2. **Alice 不想依赖一个在线的可信第三方 Online-TTP**，因为 Online-TTP 的引入会产生费用，同时降低了自己和 Bob 的通信效率。更重要的是，一旦 Online-TTP 遭受攻击，给通信双方带来的损失都是不可挽回的。

   Bob 的想法与此相同。

3. **Alice 不想依赖一个离线的可信第三方 Offline-TTP**，因为 Offline-TTP 的引入会产生费用，同时因为 Offline-TTP 不可能一直保持在线，更加降低了自己和 Bob 的通信效率。

   Bob 的想法与此相同。

因此，Alice 和 Bob 都需要这样一个这样的系统：

1. 在双方互不信任的前提下，能够实现公平的合同签署过程；
2. 摒弃对第三方的依赖。

1.3 作品介绍

基于以上情景，我们开发出了公平异步合同签署系统，命名为“**FairContract**”。在这一系统中，我们主要提供以下三个功能：

合同文本到公私钥的等价转化。

这里我们通过前端用户的合同文本和参数的输入，经过加密算法转化成合法的 ECDSA 密钥对，进而构造某交易使得其输出脚本包含该公钥生成的收款地址，可以证明参与方对该交易单的签名就等价于对合同文本的签名。

安全验证计算。

这里我们需要确保用户双方所提交的合同文本的一致性以确保签名成立。借助零知识证明和隔离见证技术，我们能够向用户证明对方提供的公钥确实是由其所需的合同文本产生而非来自一个新的文本。

签名交换。

这里是我们系统的核心功能，根据双方原始输入所产生的一系列公私钥和地址，生成整个流程所需的比特币交易单并发布。最后，由双方客户端启动交换签名过程，交换流程在比特币系统中运行实现。在交换过程中，系统提供对交易过程进展的查询与简单提示，帮助用户了解当前交易进展与所需操作。

1.4 特点描述

“FairContract”能够使两个互相不信任的用户完成对一份合同的签署。该系统具有以下特点：

去中心化：基于比特币交易网络，交易过程无第三方的参与，协议的执行过程由双方控制，具有很强的安全性

公平性：即诚实者（交出签名的一方）不会有任何损失，失信者（收到对方的签名但是不交出自己签名的一方）会受到巨额惩罚；

异步性：即双方不必同时交出签名，允许双方交出签名之间有一定时间差；

可撤销性：即双方都可以在系统执行的前半阶段退出，若如此做，系统终止，双方都没有损失，双方都拿不到对方的签名；

去中心化：即全程没有第三方的参与，双方的签名走向与资产走向由自己决定，执行过程由分布式的区块链网络系统保证。

值得注意的是，系统不具有任何干涉行为与引导行为，用户的签名交换与保证金的流向绕开系统，完全在比特币系统里面进行，完全排除了中心攻击的风险；另外，公认安全的比特币网络也让中间人攻击者望而却步。总之，和一般的公平交换系统相比，FairContract 具有更高的安全性能。

1.5 应用前景

在去中心化的比特币系统的潮流下，FairContract 使得去除第三方的公平交换成为可能，这无疑让公平交换的安全性提升了一个台阶，让用户不必担心中心攻击的问题，完全为自己的信用负责，能够满足客户的需求；

另外，FairContract 能够作为一个可以扩展开发的平台，为一些大型协议的实现提供去第三方公平交换的基础，从而在安全多方计算、可证明安全、多方认证等领域进行发挥，我们相信，传统的方案与新兴的比特币分布式可信网络的充分结合将会是未来的发展趋势。

第二章 作品设计与实现

2.1 FairContract 系统总体设计方案

根据让合同签署（即签名的交换）真正实现去第三方的目标，我们将 FairContract 客户端与去中心化的比特币系统相连接。但是由于比特币系统脚本对一笔交易的输入与输出的限制，我们不能直接将双方对合同的签名以比特币交易的形式发送出去，所以我们整体的思路是：双方根据合同文本与一个特定的随机数等价转换为一个比特币账户的公钥和私钥，从而将一方对合同的签名过程等价为对输出地址中含有转化后公钥的交易单签名；随后构造比特币交易网络，通过特殊交易的签名与广播来实现异步的合同签署过程（即签名交换过程）。

根据上述思路，我们给出一个完整的异步合同签署过程（假设双方分别为 A 和 B）：

1. A 和 B 预先通过其他渠道确认要签署的合同文本内容；
2. A 和 B 分别进行从合同文本到一个 Bitcoin 账户公私钥的等价转换；
3. A 和 B 分别向对方在步骤 2 中得到的公钥和私钥（不会将私钥告知对方）进行安全证明，确保其与合同的等价性；
4. A 和 B 构建一个比特币交易网络，并用自己原有的比特币账户和对其中的关键交易进行签名，完成对合同的签署。

2.1.1 系统功能模块

基于前文对一个完整的异步合同签署过程的描述，我们 FairContract 设计了三个主要的功能模块，如图 2-1 所示。

这三个功能模块分别为：

从合同文本到一个 bitcoin 账户公私钥的等价转换。此功能模块为后面的构建比特币特殊交易网络做好了准备，也是本系统实现一种新型签名方案的模块；

安全验证计算。此功能模块使得双方都能认同对方生成的公私钥与合同文本之间的一致性，为整个签署过程提供了安全保障；

签名交换，即特殊交易的构造与广播。此功能模块通过构造并广播特殊交易实现双方上传签名与赎回保证金的关联，从而完成异步的合同签署过程，是整个系统的核心部分。

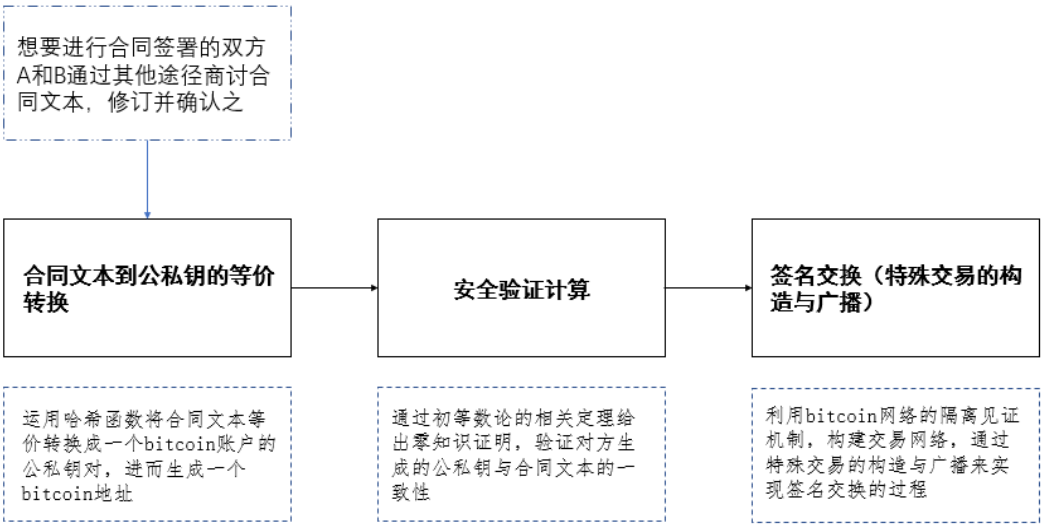

图 2-1　系统的功能模块及主要原理

2.1.2 系统结构

系统分为三层结构：底层的本地数据库，客户端的后台以及客户端的前端。它们独立负责的功能分别为：

本地数据库。存储用户由合同文本等价转换而来的公钥与私钥，以及代表用户信息的随机数等信息；存储对方对特殊交易的签名（即对方尚未生效的对合同的签名）；存储用户的用户名和密码；存储用户的历史交易内容。

客户端后台。进行与合同签名等价转换、安全验证计算的相关运算。

客户端前端。以界面的形式让用户对每个环节进行操作。

这三层结构之间，以及与比特币网络的交互关系图 2-2 所示。

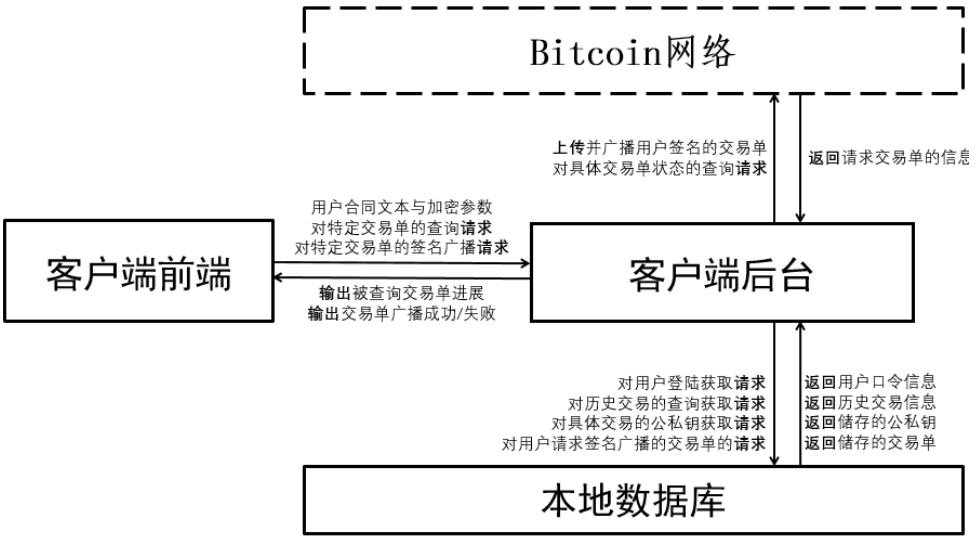

图 2-2　系统的结构以及其间的交互过程

2.2 系统的实现原理

2.2.1 “等价转换”功能模块协议

由于比特币系统脚本的安全性限制，我们不能直接将双方对于合同的签名通过纯粹的比特币交易上传到区块链上，区块链上传递的只有交易单，即比特币账户的签名。（注意：比特币账户的签名和双方对于合同的签名是两个概念）所以我们通过密码学的相关技术，将双方对于合同的签名一一映射为在某笔交易的交易单上的签名，即用合同和随机数转化为比特币账户的公钥和私钥。这样以来，输出脚本含有该地址公钥的交易被签名，便意味着对合同的签名开始生效。

我们假定参与签名交换的双方分别为 Alice(以下称为 A)和 Bob（以下称为 B），相应的合同文本分别为M_A, M_B（B 需要的是 A 对M_A的签名，A 需要的是 B 对M_B的签名）。由于双方的密钥生成过程都是完全相同的，因此我们只给出 A 的视角，将合同M_A转化为比特币网络中某密钥对 (sk_A, pk_A)。B 视角的过程可以同理得到。

任意一方（假设为 A）生成相应的密钥对(sk_A, pk_A)的协议是：

1. A 选取比特币系统加密使用的椭圆曲线 secp256k1：$y^2 = x^3 + 7 (mod\ p)$的生成元$G(\mathrm{x}_G, y_G)$,其阶数为q，再随机选取两个大素数 p_A, q_A,计算$N_A = p_A \cdot q_A \cdot q$，则$\varphi(N) = (p_0 - 1)(q_0 - 1) \cdot \varphi(q)$，这里要保证$N_A$至少是 2048 位；

2. A 选择随机数r_A，计算合同M_A的哈希加盐值$h_A = H(M_A, r_A)$;

3. A 计算得到私钥$sk_A = [h_A^{\frac{1}{3}} \ (mod \ N_A)] \ (mod \ q)$，公钥$pk_A = sk_A \cdot G$；

4. A 将$(M_A, r_A, h_A, N_A, q, G, pk_A)$作为公开参数传递给 B，而$(sk_A, p_A, q_A)$则需要保密。

比特币网络中的 ECDSA 签名要求私钥 sk 小于特定值(0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141)，在实际生成中，我们利用比特币所用标准 ECC 曲线，产生 ECDSA 公钥，并零知识证明 ECDSA 私鈅与上面三次根在模 ECC 阶下相等完成。

以上，$(M_A, r_A, h_A, N, p, g, pk_A)$都作为公开参数可以共享给 B，而$(sk_A, p_0, q_0)$需要 A 自己保存。

协议的数据传递过程如图 2-3 所示.

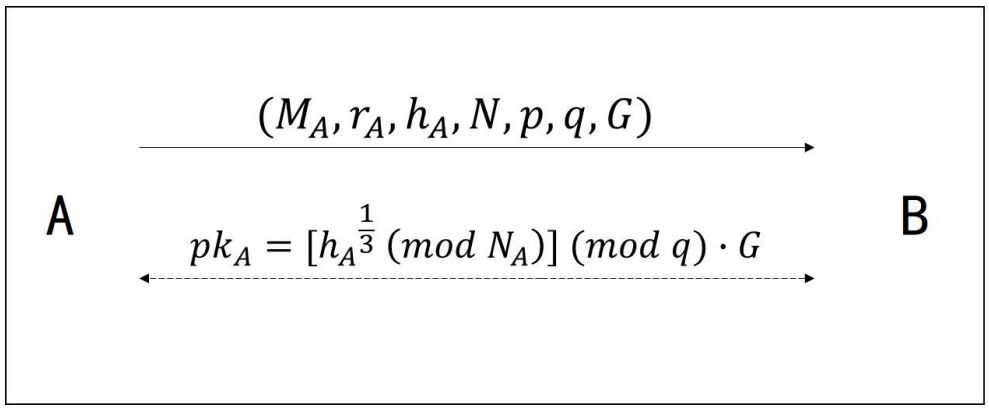

图 2-3　等价转换协议的数据传递

2.2.2 “安全验证计算”功能模块协议

在这里仍然以 A 的视角，她需要让 B 相信 A 的比特币账户公私钥确实是由合同M_A生成的，而不是由M'_A生成的。在安全验证计算的过程中，A 不能将私钥sk_A透露给 B，因此要用到零知识证明的方法。

下面给出本协议的方案。

1. 设$x = sk_A = [h_A^{\frac{1}{3}} \ (mod \ N_A)] \ (mod \ q)$,A 计算

$$\begin{cases} X = x \cdot G \\ Y = x \cdot X \\ \quad = x^2 \cdot G \\ Z = x \cdot Y \\ \quad = x^3 \cdot G \end{cases}$$

2. A 选择一个随机数$k \epsilon Z_q^*(1 \leq k \leq q-1)$, 计算$s = H(X, Y, Z, k \cdot G, k \cdot X, k \cdot Y)$.

3. A 计算$t = k - xs(mod\ \ q)$, 并将(X, Y, Z), (s, t)发送给 B.

接下来，B 只需要按照典型的零知识证明的方法操作即可，这样，A 就像 B 证明了密钥对(sk_A, pk_A)确实是由合同M_A生成的，而不是由M'_A生成的。

2.2.2 “签名交换”功能模块方案

2.2.2.1 方案概述

此前，我们将合同文本 M 等价转化为某 ECDSA 的公钥 pk。如果将 pk 的哈希值置于比特币赎回交易的输出脚本中，那么参与方对新交易单签名的交易被发布并被确认就等价于对合同 M 的签名。基于以上考虑，我们构造出相应的签名交换模型：

交换双方 A 和 B 首先各自支付 x BTC 作为交换过程的保证金。在交换过程中，如果 A 遵守协议即在约定时间 T 内提供有效的合同签名就可以将属于自己的 x BTC 赎回；否则作为惩罚，A 将损失 x BTC，2x BTC 被全部支付给 B，交换中止。在第一种情况下，如果 B 同样遵守协议即也在约定时间 T 内提供有效的合同签名就可以将属于自己的 x BTC 赎回；否则作为惩罚，自己的 x BTC 被支付给 A。

A 和 B 与比特币系统之间的合同、保证金与其他参数的交互如图 2-4 所示：

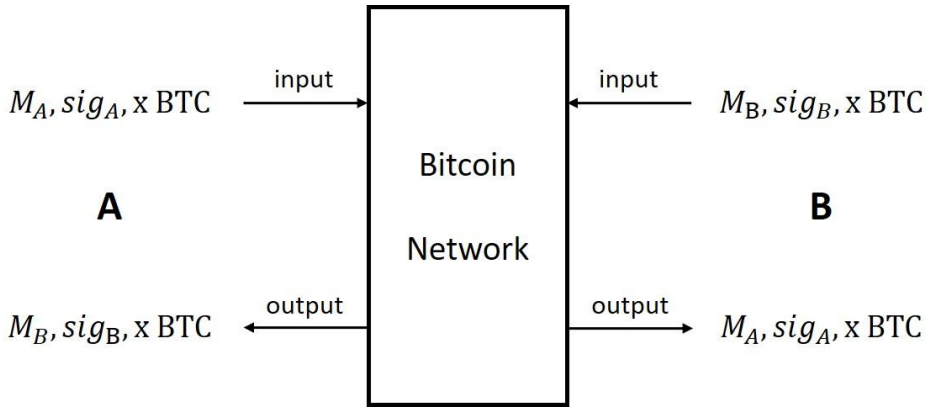

图 2-4　数据交互示意图

2.2.2.2 交易网络的总体框架

借助隔离见证[3]技术，我们给出签名交换的比特币交易网络的总体框架，如图 2-5 所示。

其中，A 和 B 各自需要准备两个比特币账户：

一是原本账户，即和需要签名的合同无关的账户，这个账户用来对每一笔交易单

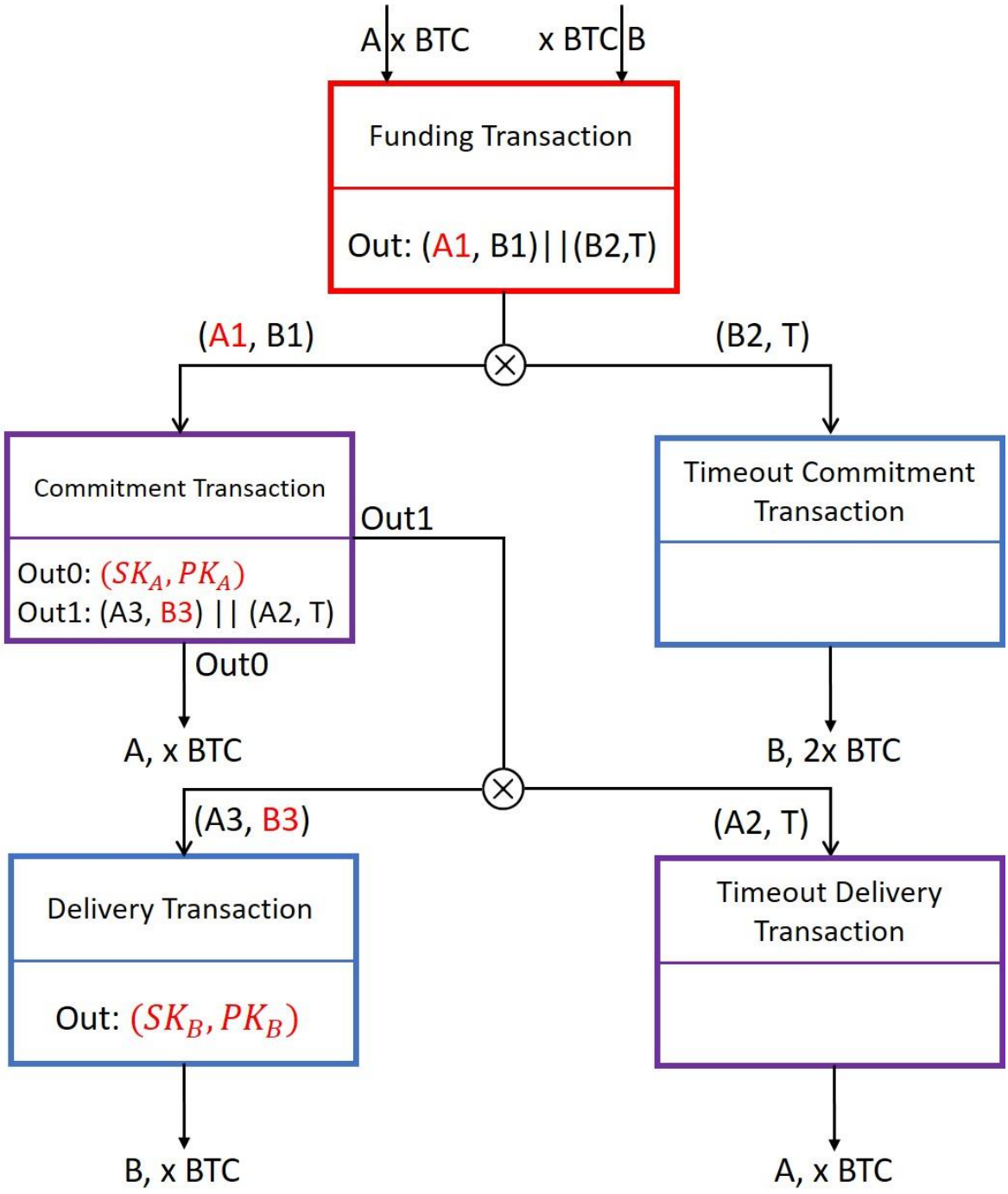

图 2-5 交易网络的整体框架

进行签名。

二是合同账户，即上文提到的由合同等价转换而来的公私钥所在账户，这个账户被写进与合同签署有关的关键交易的输出脚本中，并作为该交易生效后保证金的输出

地址。

2.2.2.3 交易网络的具体结构

我们构建的交易网络分为五笔交易，下面具体介绍每笔交易的内容及作用。

Funding Transaction （资金入块交易，以下简称 F）

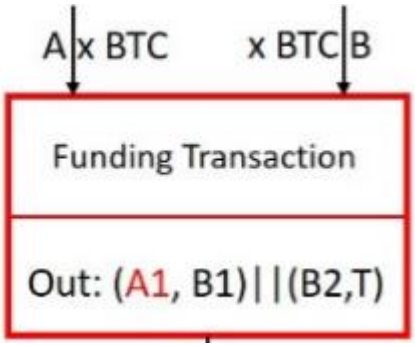

图 2-6 Funding Transaction

Inputs: 1.A x BTC 2.B x BTC

Outputs: (A1,B1) 2x BTC OR (B2,T) 2x BTC

如图 2-6 所示，Funding Transaction 由 A 和 B 共同生成，双方各提供 x BTC 的输入，其输出为 2x BTC。在 T 的时间内，可以通过 A 和 B 的共同签名赎回；在 T 时间后 B 可以通过自己的签名赎回。这意味着，如果 A 想要赎回自己的保证金，就必须在 T 时间内发布包含自己数字签名的子交易；否则作为惩罚 x BTC 将被支付给 B。

Commitment Transaction （承诺交易，以下简称 CT）

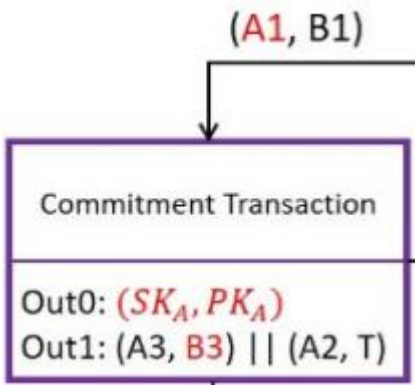

图 2-7 Commitment Transaction

Inputs: (A1,B1) 2x BTC

Outputs: 0. A x BTC 1. (A3,B3) x BTC OR (A2,T) x BTC

如图 2-7 所示，Commitment Transaction 由 A 生成，但是由于其输入来自于父交易 F，因此该交易在广播之前需要先交由 B 进行签名。交易单的输出包括两部分：支付给自己的 x BTC 的 Out0，用双重签名和时间 T 重新锁定的 Out1。

Timeout Commitment Transaction （承诺超时交易，以下简称 TC）

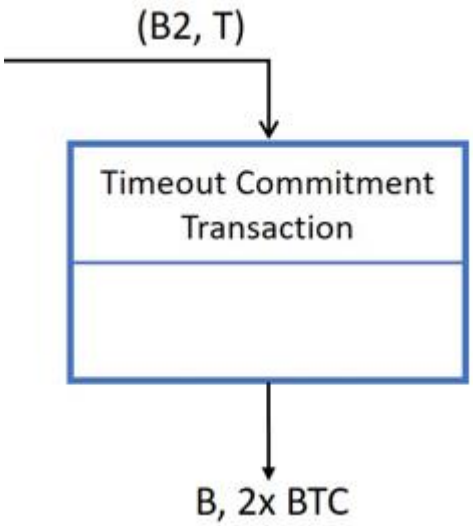

图 2-8 Timeout Commitment Transaction

Input: (B2,T) 2x BTC

Output: B 2x BTC

如图 2-8 所示，Timeout Commitment Transaction 由 B 生成，并在父交易 F 被确认 T 时间之后进行广播才会被确认有效。该交易主要用于对 A 的行为进行限制，如果 A 在时间 T 内未发布上述交易 C 即没有提供数字签名，那么该交易将得到确认，作为惩罚将 A 和 B 的保证金全部支付给 B，交换过程因为 A 的行为不端被迫中止。不过由于 x BTC 远大于该数字签名本身的价值，因此 A 更倾向于在 T 时间内公布有效的数字签名。

Delivery Transaction （传达交易）

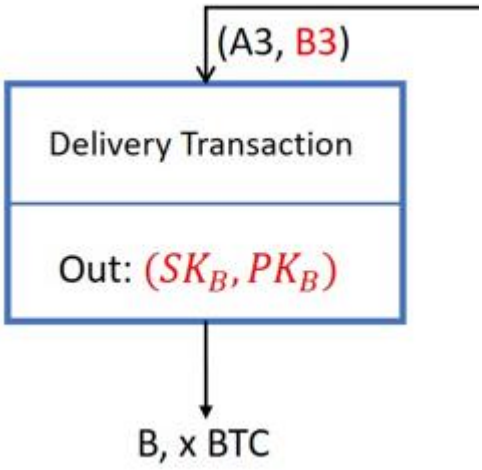

图 2-9　Delivery Transaction

Input: (A3,B3) 2x BTC

Output: B 2x BTC

如图 2-9 所示，Delivery Transaction 由 B 生成，但是由于其输入来自于父交易 Commitment Transaction，因此该交易在广播之前需要先交由 A 进行签名。该交易只有一个输出 Out0，指向合同M_B转化对应的公钥地址，相应于支付 x BTC 支付给 B。

Timeout Delivery Transaction （传达超时交易）

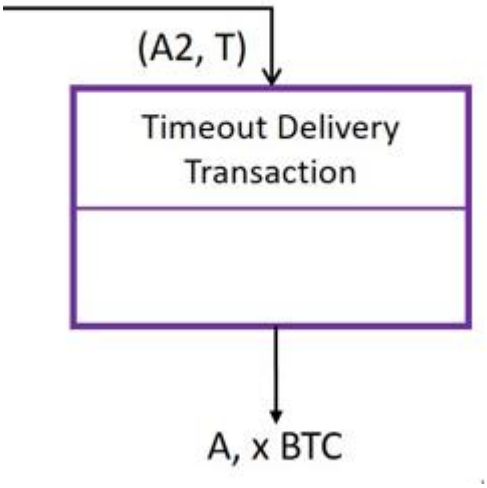

图 2-10 Timeout Delivery Transaction

Input: (A2,T) x BTC

Output: A x BTC

如图 2-10 所示，Timeout Delivery Transaction 由 A 生成，并在父交易 C 被确认 T 时间之后进行广播才会被确认有效。该交易主要用于对 B 的行为进行限制，如果 B 在时间 T 内未发布上述交易 D 即没有提供数字签名，那么该交易将得到确认，作为

惩罚将 B 的保证金支付给 A，交换过程因为 B 的行为不端被迫中止。不过由于 xBTC 远大于该数字签名本身的价值，因此 B 更倾向于在 T 时间内公布有效的数字签名。

前文已经明确，任意一方一旦使用自己的私钥对输出脚本中包含生成的公钥的交易单进行了签名，就意味着该方对合同签了名。所以，在我们构建的签名交换框架中，Commitment Transaction 被 A 签名，意味着 A 对合同的签名生效，Delivery Transaction 被 B 签名，意味着 B 对合同的签名生效。

值得注意的是，在我们构建的交易框架中，A 和 B 并不是对称的关系。A 相对于 B 来说，需要提供签名的时机会更早，相应地，A 拿回自己的比特币的时机也较早，也就是说，A 是率先的执行者。但是，这仅仅是先后顺序的差别，丝毫不影响整个方案的公平性。

2.2.2.4 签名交换的准备过程

隔离见证技术的应用使得我们可以在交易 F 被广播之前生成消费其输出的交易 C 和 TC，并对该交易单进行签名。当然，交易 C 和 T C 尽管得到所需要的签名也无法被网络节点确认，因为其父交易 F 尚未得到确认，那么该交易单的输入部分就无法被认定为合法。利用这一点，我们制定下面的签名交换准备步骤：

第一步：B 提供 x BTC 的交易输入，发送给 A;

第二步：A 提供 x BTC 的交易输入，生成交易 F 发送给 B；之后根据交易 F 生成交易 C，也发送给 B;

第三步：B 在判断交易 F 和 C 合理后，生成 D 发送给 A，并生成交易 TC 并签名；

第四步：A 对交易 D 提供签名，发送给 B；并生成交易 TD 并签名；

第五步：B 对交易 F 和 C 提供签名，发送给 A；

第六步：A 对交易 F 进行签名并广播出去；

在父交易 F 生成阶段，如果有一参与方不愿意提供 x BTC 的输入，那么后续交换过程便可以取消；但是提供 x BTC 的参与方并不会积攒相对的劣势，因为对该交易单 2x BTC 的赎回仍需要双方签名才能生效。因此，绝大多数情况下，该交易单得以顺利生成。在子交易 C、TC、D、TD 生成阶段，A 和 B 根据其父交易的输出脚本设定和交换约定生成相应输入和输出的交易单。其中交易 C、D 都需要双重签名，如果 A

或者 B 的交易单不符合协议约定，对方可以拒绝进行签名。

在前四步完成的基础上，双方可以对父交易 F 进行签名。交易 F 虽然由 A 生成，但任何一方不提供签名将使得该父交易无法生效，因此双方都不会遭受任何损失，不存在优势、劣势的区别。为了使得交换过程尽快开始，绝大多数诚信参与方都会尽快提供签名。在父交易签名完成后，就可以将该交易广播出去，经由网络节点进行确认进入区块链。

2.2.2.5 签名交换的实施过程

在签名前期准备完成后，父交易已经进入区块链主链，而参与方 A 的交易 C 得到了 B 的签名；B 的交易 D 也得到了 A 的签名。交换过程如下：

1.如果 A 在时间 T 内签名使得交易 C 生效，A 将得到 x BTC；否则，在时间 T 后交易 TC 被确认，B 得到 2x BTC，A 因为未及时公布签名而承受 x BTC 的惩罚，交换过程中止

2.在交易 C 被确认后，A 广播交易 TD。如果 B 在时间 T 内签名使得交易 D 生效，B 将得到 x BTC，交换过程正常结束；否则，在时间 T 后交易 TD 被确认，A 得到 x BTC，B 因为未及时公布签名而承受 x BTC 的惩罚，交换过程中止。

A 提供对交易 C 签名的同时也就代表向对方公布了数字签名。不过该交易的生效不仅需要提供有效的签名，还必须在规定的时间限制 T 内进行广播。当且仅当该数字签名在时间 T 内得到有效验证，交易 C 才会稍后得到确认，也就是向pk_A对应的地址支付 x BTC，同时重新建立需要双重签名才能使用的输出 Out1。只有 A 根据合同M_A得到了密钥对(sk_A,pk_A)，A 可以随时消费输出 Out0，意味着 A 已经得到 x BTC。X BTC 远远大于其数字签名本身的价值，如果 A 不遵守约定，作为惩罚 2x BTC 将被全部支付给 B，A 将损失比起数字签名价值更高的资金。在这种前提下，A 绝大数情况下都会遵守之前的交换约定。

在 A 按照约定在时间 T 内公布自己的有效签名后，B 在时间 T 内可以通过发布交易 D 取回自己的保证金。与交易 C 相同，交易 D 输入也需要提供双重有效签名，但是 A 之前就已经在准备阶段提供了自己的签名，因此这里 A 无法影响 B 广播交易 D。如果 B 没有在有效时间 T 内提供用来交换的数字签名，作为惩罚，x BTC 将被支付给 A；如果 B 按照约定及时公布了自己的数字签名，那么 B 就可以赎回自己的保

证金。而 x BTC 又远远大于数字签名本身的价值，B 不遵守约定将会受到比其数字签名价值更高的损失，在这种情况下，B 绝大数情况下都会遵守之前的交换约定。此时，签名交换完成。

2.3 FairContract 客户端的软件实现

2.3.1 开发工具的选择

首先我们的目标在于完成一个功能健全，可以方便用户双方通过独立操作实现协议签名公平交换过程的 PC 客户端。开发的过程从实现基本的比特币交易开始，逐渐完善，加入创意创新部分。一面开发，一面完成理论验证过程，在初期完成命令行工具，而后加入图形界面接口，使工具的使用平民化。

关于一些工具的选用[4]如下：

1.**开发语言的选择：Python**。考虑到项目工程最终应该具有不断壮硕可拓展的潜力，并且在开发初期便于时间精力有限的我们实现最基本的功能，我们选择 Python 作为开发语言。Python 相比 C++，Java 等高级语言具有交互便捷，语义更容易理解的优势，跨平台特性优良，使包管理和发布更加便利。

2.**界面制作工具的选择：PyQt**.选用 PyQt 作为 UI 图形库使得我们的页面设计工作事半功倍。

3.**数据库的选择：sqlite**.区块链技术涉及大量的序列，字符的运算与存储，考虑到产品最终的用户友好性，将非对称密钥，地址，签名等长序列以标签的形式存入 sqlite，方便客户的调用，免去记忆与输入的麻烦。

4.**测试环境的选择:blockcypher**.产品测试和使用环境依赖 blockcypher 提供的 API[5]接口，如此以来，使用客户无需同步 200GB 的区块数据，同时一定程度降低我们的开发难度，将注意力集中于实现逻辑功能和测试，而非网络通信。

我们的作品是一个从理论到实现的彻底创新，因此有别于以往的创作，测试与验证并行，在验证的基础上开发完善，得以实现。所以实现难度更大，创新的意义非凡。

2.3.2 客户端前端主界面实现

图 2-11 是我们主界面的架构设计，在用户登录进主界面后，有以下几个模块：

查询历史合同签名交换。数据库中会保存历史交易记录，用户可以查看历史账单及其最终状态。

查询当前合同签名交换进度。可以获取交易单信息和当前交易进度，以及操作剩余时间等信息，同时可以对当前交易单进行简单操作。

发起一次合同签名交换。这里用户填写交易对象与合同以及合同加密等信息，后台进行具体运算并生成公私钥。

存在合同签名交换申请。用于查看他人对用户的交易申请，如果选择接受则用户将上传自己的合同文本及参数。

图 2-11　主界面的架构示意图

首页界面如图 2-12 所示：

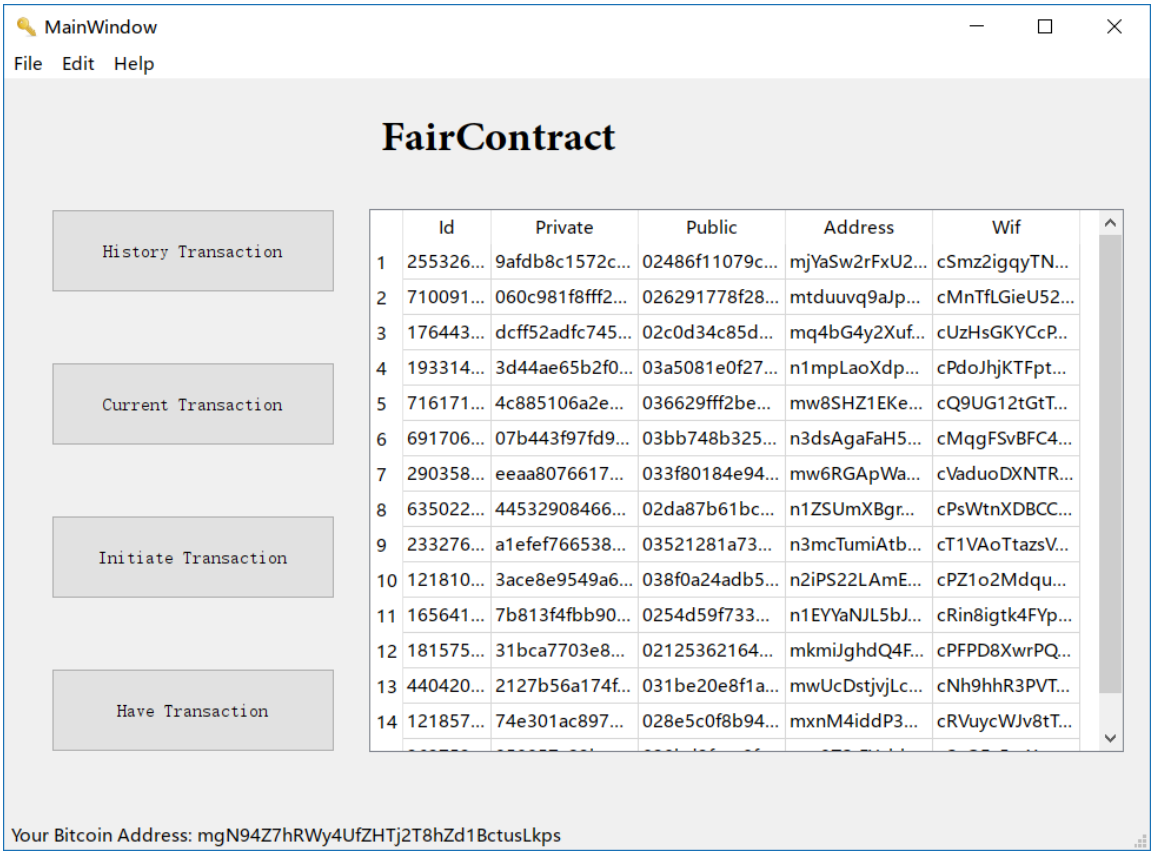

图 2-12　用户登录后的首页界面

2.3.3 数据库后端与后台运算部分的实现

我们采用 sqlite 作为建立数据的工具。前文中已经描述了数据库的功能，据此建立了以下几个表格：

sk 表用来存储用户由合同文本等价转换而来的公钥与私钥，以及代表用户信息的随机数等信息，如图 2-13 所示；

rowid	content_id	content	tran_pubkey
(empty)	*(empty)*	*(empty)*	*(empty)*
1	1824716216	\nAppointments\n Possibilities range from: showing up, making a telephone call to set up an appointment, to using an intermediary. But, essential to know how to procedure to follow in setting up appointments.\n\nQuestion: How essential are appointments in the US? In the US, appointments can be made quite casually and on short notice. Possible, even, to do a \xe2\x80\x9cwalk-in\xe2\x80\x9d appointment. And, if the person is \xe2\x80\x9cfree\xe2\x80\x9d the appointment can take place right then. I experience this quite often with sales people seeking my business here in the School of Business. However, this is not true in many other cultures.\n	ddf6621aa8c70ab65fff236ea4a969b600c9520b3
2	1850657452	Examples:\nIn Latin America. Appointments must be made a least one month in advance by mail or telephone and followed up one week before hand. Contacts should also be made as high up in the organization as possible and be done so through a \xe2\x80\x9cconnection\xe2\x80\x9d if possible.\n	0f0b4207e2f60ffa0ea35c562a415769f936a2a21
3	427561984	In China, essential to set up appointments before you even come to the country.\n	cab22420666abb718b2861e4ec362fcd99c98be7
4	1158561399	In Saudi Arabia, need to have a sponsor who sets up the appointments for you.\n	15db88ba09b5334779e734c066ad8a4c8bf612e
5	2122773115	In Japan, while a connection (who is not part of either your or the other company) is often very useful as a means of introduction, a personal call is the best way to make an appointment. A letter may not even be answered.\n	3701b1b5994a586bb18c7089e372295e51823cd

图 2-13　sk 表

sig 表用来存储对方对特殊交易的签名（等价为对合同的签名），如图 2-14 所示；

rowid	our_raw_tran_id	unsigned_sequence	signed_sequence
(empty)	(empty)	(empty)	(empty)
1	2024062331	0100000001000000000000000000000000000000000000000000000000000000000000000000000000	0100000001000000000000000000000000000000000000000000000000000000000000000000000000
7	935183685	01000000018c6125a352dfc84a98b7fcf78f8eae6e9e1e15733dbc2d5b6262615095	01000000018c6125a352dfc84a98b7fcf78f8eae6e9e1e157
6	765592637	0100000001a1b7173df848001f783e99212490b34ba2a59e818b5e815354dd5350f	0100000001a1b7173df848001f783e99212490b34ba2a59
5	703300985	0100000001a98d2fde6c21290487e178ad930e41ecd7512077769355838172dbf65	0100000001a98d2fde6c21290487e178ad930e41ecd7512
9	616105385	0100000001c838044a55e46aeb90ba1ec0006683655dab84f43bf2c91f001830f4a5	0100000001c838044a55e46aeb90ba1ec0006683655dab8
3	1787741753	0100000001eb3eb633bd37d6aed53b6ef2e3e49deccf8c0c904771528215454ee7c	0100000001eb3eb633bd37d6aed53b6ef2e3e49deccf8c0
8	1656305597	0100000001eb50b3b0f246defe91db04f85df4dad5360700210fd4de072ebc4bd68	0100000001eb50b3b0f246defe91db04f85df4dad536070
4	1023192363	0100000001f7a015b40e570c9eeb1a5309b128355b167d369b33c874ae20b27cd90	0100000001f7a015b40e570c9eeb1a5309b128355b167d3
2	994161761	01000000fd4101c5e227956fa19377c28d907c03b4ac16bd7433d5ae47a2da715d0	01000000fd4101c5e227956fa19377c28d907c03b4ac16bc

图 2-14　sig 表

account 表用来存储用户的用户名和密码，如图 2-15 所示；

rowid	id	private	public	address
(empty)	(empty)	(empty)	(empty)	(empty)
1	1433556946	58f4a62bb6bd27b27f32a25e7b9ce80a1e505e06fc15b070e9937694e9bc25d1	02f995822eb6c57fc585b6754aad18be2ca8fb7ebbe6c5bdadfbcf350daa1fe6a1	mreM7a
2	1034537049	19ffffe3c038f934c78e8ab94dd984f92357b86eaa67870606a3b41ed09a0b4c	021069185ba40aff05e53410e0ffa45259d4a73ce4fddaebb21b1eb4a591764a65	mkGB1c
3	1195704967	0f6fbef70d4b8661fe1db2c051df34041fe4a824eaa4af92250cbd9092e4ef0e	03e2731e80b55b6e6f781e15155e89660b4fa167f64bb9fe8496c1d81afb0bf745	mi8UCFh
4	2103903753	3c7b3b2c7d211ea72666265be78c2f2efcccb385d6139a02416f1cec4536b015	033752d99d391190c1c082a5bbd099e8979435b2aaa8bf7671d3fd02bcac3e986c	miGoT4h
5	2064985877	6b103177146b6df39ff53f1c8db1d475aff8d5f6821b50dc2e41eca81691d5ec	0343bb7f69c2b6184ce2e0948cb44607bbf921717cdf1b61ebbdcb956b38054696	mm1bRa
6	271229769	56bbdbc06e007cce2f647b697395cacdfa2dc602383a61f2a4b6523990afd4bb	021146b87ce11d643cace8ff5f241f9b38bb90011f2a4be720fe068db4c18fb24c	mim1juc
7	814667955	4bbe9c5cedb1e8a065fe90d130167ce309c22992bc1afe8bc14ed12ff924f07d	030a431b787ba187042ad2f755abe6b542208775151bffbe711edc2867825b4319	mnc3Zc
8	1840903857	f3f5256438c8ef376472c368dbc982ba11bf27d7cf1d836ae1e3234840daca92	02895bbdcde8ffe1e1c596e448f764c8cb600a24f7194527655258f80b453a7c1c	mqX3br

图 2-15　account 表

history 表用来存储用户的历史交易内容，如图 2-16 所示：

rowid	sequence	txid
(empty)	(empty)	(empty)
1	1242539480	77b10dc0d8fea0a83fba36a8bd95a487191a4498996ef82216b93d89541f3f2b
2	337825676	cd048168303d9f0036abf92fb1d47d117efd4c0e243a97cde2c21075d6a2cd20
3	1953731307	c20859af675e55392e60bfa2d7a2b9131092e6636a8cf407c17ec476eae82a77
4	236515360	0cbd9620da46d1c77ca37daa9ff2649610425892dc5d617a740d129d83f79fab
5	983066464	26f904c3c6f33796c641d28e2571ccd03d848bd0b8798d6d796b6de7ff64d6d9
6	2072972727	65e816e1ea486e3384df21674fe94ff47a0ea6f2fffb9d11d7318ead8f56603c
7	2041976102	1e921e0c124a56c187679f5bcc1eeca5ec17bd777bb2dd20b4b2c0d5926495b
8	796505848	5f73fcbfd831718392e66f1f91d58915b3ffd76924c6c88e174c09c5dc4670b8
9	1095958496	b8fedee984cbe6c2171bcda214840a41d5e735de619114c8f18c00806c7b9326

图 2-16　history 表

客户端后台关于交易的构造与上传的实现将在 2.3.4 中详细介绍，剩余的功能便是进行相关运算（等价转换与安全验证），已经按照前文的协议实现。

图 2-17 是数据库表格建立的部分代码：

```
# -*- coding: utf-8 -*-
####here are works for initiation
#####
#####
sqlitedbname='accounts.db'
import sqlite3
from os.path import exists
import os
def setupsql():
    try:
        if not exists(sqlitedbname):
            fp = open(sqlitedbname,'w')
            fp.close()
        conn=sqlite3.connect(sqlitedbname)
        c=conn.cursor()
        sql="create table account(id int primary key,\
                                  private varchar (64),\
                                  public varchar(65),\
                                  address varchar(34),\
                                  wif varchar(60))"
        #actually, don't forget the real len is 64 33 34 towards these 3
        c.execute(sql)
        conn.commit()
        conn.close()
    except:
        print "table exists already!"


def delete_sql_table():
    try:
        conn = sqlite3.connect(sqlitedbname)
"oursetup.py" 43L, 1131C
```

图 2-17　数据库表格的建立部分代码

2.3.4 交易网络的实现

2.3.4.1 交易单的构造实现

本系统在测试过程中，将后端与比特币的测试网络相连接，交易单使用了 BlockCypher 的交易单格式，具体组成如图 2-18 所示：

Field name		Type (Size)	Description
nVersion		int (4 bytes)	Transaction format version (currently 1).
#vin		VarInt (1-9 bytes)	Number of transaction input entries in *vin*.
vin[]	hash	uint256 (32 bytes)	Double-SHA256 hash of a past transaction.
	n	uint (4 bytes)	Index of a transaction output within the transaction specified by *hash*.
	scriptSigLen	VarInt (1-9 bytes)	Length of *scriptSig* field in bytes.
	scriptSig	CScript (Variable)	Script to satisfy spending condition of the transaction output (*hash*,*n*).
	nSequence	uint (4 bytes)	Transaction input sequence number.
#vout		VarInt (1-9 bytes)	Number of transaction output entries in *vout*.
vout[]	nValue	int64_t (8 bytes)	Amount of 10^{-8} BTC.
	scriptPubkeyLen	VarInt (1-9 bytes)	Length of *scriptPubkey* field in bytes.
	scriptPubkey	CScript (Variable)	Script specifying conditions under which the transaction output can be claimed.
nLockTime		unsigned int (4 bytes)	Timestamp past which transactions can be replaced before inclusion in block.

图 2-18　一份比特币交易单的构成

除了交易单的 id 以及与锁定时间有关的 nLockTime，一个交易单的主要组成部分便是输入部分与输出部分。其中，我们可以控制的部分除了输入与输出的比特币金额，便是至关重要的输入脚本 sigscript 与输出脚本 pubkeyscript，这是我们使用隔离见证技术构建多重签名交易的基础。

比特币的输入脚本 sigscript 与输出脚本 pubkeyscript 为我们提供了基于堆栈的脚本语言以供编写。前文中设计的交易网络较为复杂，每一笔多重签名交易的实现都需要脚本语言的堆叠。

下面给出交易网络中实现每笔交易单的脚本语言：

1.Funding Transaction（图 2-19）

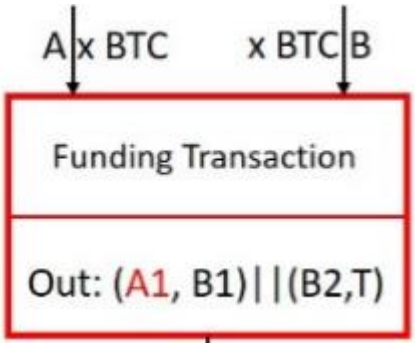

图 2-19　Funding Transaction

PubKeyScript：

OP_DUPOP_HASH160<HASH160(RedeemScript)>OP_EQUALVERIFY

其中<RedeemScript>：

OP_HASH160<HASH160(hA)>OP_EQUALVERIFY

OP_IF

OP_2<B1PubKey><A1PubKey>OP_2OP_CHECKMULTISIG

OP_ELSE

<Ttime>OP_CHECKSQUENCEVERIFYOP_DROP<B2PubKey>OP_CHECKSIG

OP_ENDIF

2.Commitment Transaction（图 2-20）

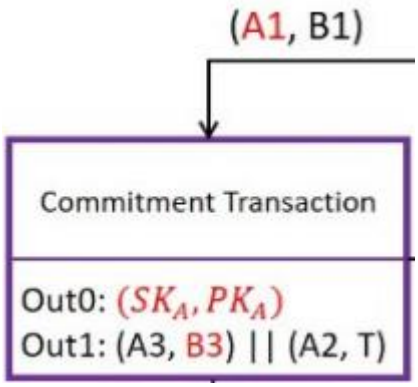

图 2-20　Commitment Transaction

SigScript：

OP_0<SigB1><SigA1><hA><RedeemScript>

其中<RedeemScript>：

OP_HASH160<HASH160(hA)>OP_EQUALVERIFY

OP_IF

OP_2<B1PubKey><A1PubKey>OP_2OP_CHECKMULTISIG

OP_ELSE

<Ttime>OP_CHECKSQUENCEVERIFYOP_DROP<B2PubKey>OP_CHECKSIG

OP_ENDIF

PubKeyScript：

Out0:

OP_DUPOP_HASH160<HASH160(pkA)>OP_EQUALVERIFYVERIFYOP_CHECKSIG

Out1:

OP_DUPOP_HASH160<HASH160(RedeemScript)>OP_EQUALVERIFY

其中<RedeemScript>：

OP_HASH160<HASH160(hB)>OP_EQUALVERIFY

OP_IF

OP_2<A3PubKey><B3PubKey>OP_2OP_CHECKMULTISIG

OP_ELSE

<Ttime>OP_CHECKSQUENCEVERIFYOP_DROP<A2PubKey>OP_CHECKSIG

OP_ENDIF

3.DeliveryTransaction（图 2-21）

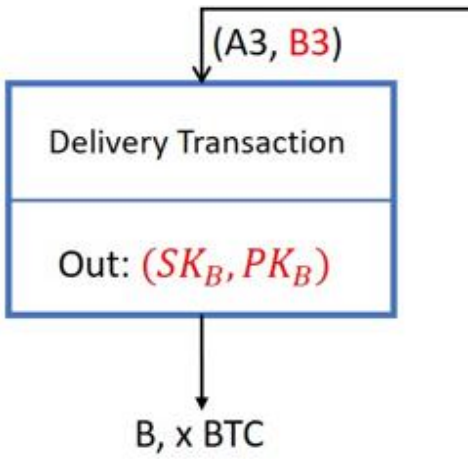

图 2-21　Delivery Transaction

SigScript：

OP_0<SigA3><SigB3><hB><RedeemScript>

其中<RedeemScript>：

OP_HASH160<HASH160(hB)>OP_EQUALVERIFY

OP_IF

OP_2<A3PubKey><B3PubKey>OP_2OP_CHECKMULTISIG

OP_ELSE

<Ttime>OP_CHECKSQUENCEVERIFYOP_DROP<A2PubKey>OP_CHECKSIG

OP_ENDIF

4.Timeout Commitment Transaction（图 2-22）

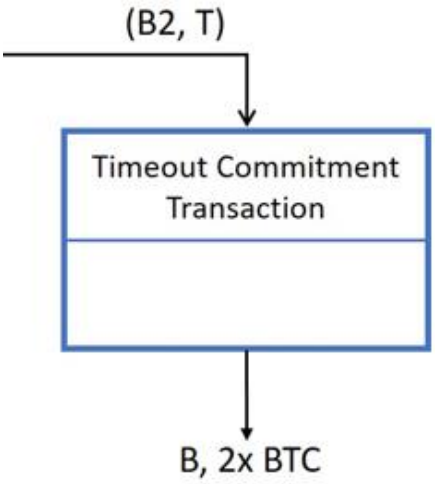

图 2-22　Timeout Commitment Transaction

SigScript：

<SigB2>OP_0<RedeemScript>

其中<RedeemScript>：

OP_HASH160<HASH160(hA)>OP_EQUALVERIFY

OP_IF

OP_2<B1PubKey><A1PubKey>OP_2OP_CHECKMULTISIG

OP_ELSE

<Ttime>OP_CHECKSQUENCEVERIFYOP_DROP<B2PubKey>OP_CHECKSIG

OP_ENDIF

5.Timeout Delivery Transaction（图 2-23）

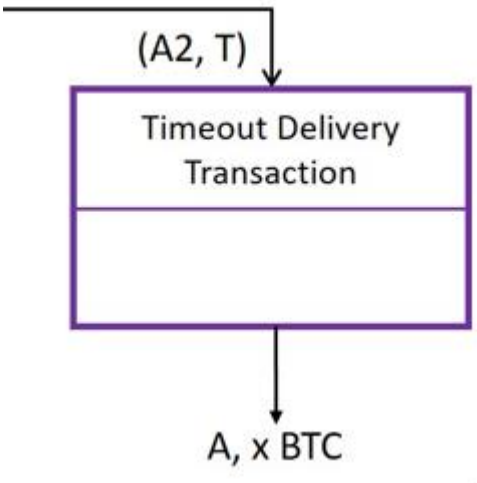

图 2-23　Timeout Delivery Transaction

SigScript：

<SigA2>OP_0<RedeemScript>

其中<RedeemScript>：

OP_HASH160<HASH160(hB)>OP_EQUALVERIFY

OP_IF

OP_2<A3PubKey><B3PubKey>OP_2OP_CHECKMULTISIG

OP_ELSE

<Ttime>OP_CHECKSQUENCEVERIFYOP_DROP<A2PubKey>OP_CHECKSIG

OP_ENDIF

2.3.4.2 交易的广播实现

在 2.3.4.1 中，我们给出了实现每笔交易的脚本语言。对一笔交易进行广播的步骤为：

1.**构建交易单**，将脚本语言以对应的 16 进制形式嵌入到交易单的相应位置，形成 json 格式的交易单。

图 2-24 是我们用已经嵌入的脚本语言构建好的一份交易单的输入与输出部分：

```
    {
        "addresses": [
            "CEztKBAYNoUEEaPYbkyFeXC5v8Jz9RoZH9"
        ],
        "age": 576,
        "output_index": 0,
        "output_value": 5000000,
        "prev_hash": "c8ea8b221580ebb2f1cabc8b40797bffec742b97c82a329df96d93121
        "script": "483045022100921fc36b911094280f07d8504a80fbab9b823a25f102e2bc
        "script_type": "pay-to-pubkey-hash",
        "sequence": 4294967295
    }
],
"lock_time": 0,
"outputs": [
    {
        "addresses": [
            "C1rGdt7QEPGiwPMFhNKNhHmyoWpa5X92pn"
        ],
        "script": "76a9145fb1af31edd2aa5a2bbaa24f6043d6ec31f7e63288ac",
        "script_type": "pay-to-pubkey-hash",
        "value": 1000000
    },
    {
        "addresses": [
            "CEztKBAYNoUEEaPYbkyFeXC5v8Jz9RoZH9"
        ],
        "script": "76a914efec6de6c253e657a9d5506a78ee48d89762fb3188ac",
        "script_type": "pay-to-pubkey-hash",
        "value": 3988000
    }
],
```


图 2-24　所构建交易单的输入与输出部分

2. **将 json 格式的交易单转为 16 进制的 hex 格式**，这是比特币系统规定的待广播交易单的格式。我们建立了 json_to_hex 函数来实现这个过程，图 2-25 是一部分：

jsontohex.py

big2little()

```
31      # add inputs to hex
32      for _, perin in enumerate(in_json['inputs']):
33          result.append(big2little(perin['txid_prev']))
34          result.append(dec2byte(perin['out_prev'], 4))
35
36          # add script_pubkey to hex
37          result.append(dec2byte(len(perin['script_sig'])/2))
38          result.append(perin['script_sig'])
39          # squence can be int
40          if isinstance(perin['sequence'], int):
41              result.append(dec2byte(perin['sequence'], 4))
42          # or be normal hex-str
43          elif isinstance(perin['sequence'], str):
44              result.append(big2little(perin['sequence']))
45
46      # add output_num to hex
47      result.append(dec2byte(len(in_json['outputs'])))
48      # add output to hex
49      for _, perout in enumerate(in_json['outputs']):
50          result.append(dec2byte(perout['value'], 8))
51
52          # add script_pubkey to hex
53          result.append(dec2byte(len(perout['script_pub'])/2))
54          result.append(perout['script_pub'])
```

图 2-25　json 转 hex 的函数

图 2-26 是由步骤 1 中的交易单 hex 化以后的序列：

"hex":
"02000000031206ce8e0eac0ffbb08bbe70bfe42074e69c726f0d66662f41aade71bb52e56b000000006a4
7304402205e3d570bf6258a2b481d4ab008b95e164046d61038c86ab876e51af912f6cee702207f691ab0f
383c7b59b8325d753e9c8b62a58a05c3605da8d14d78be8ee73bbfa012102e33a7c5c2a1a16e56683cfc2
aa4a289835e4a4b1ed440875fa2b9e672e0cbb95feffffff1206ce8e0eac0ffbb08bbe70bfe42074e69c726f0d
66662f41aade71bb52e56b010000006b4830450221008cc698f1d53f30afc720786c9d8402a4f7238ecfe0f
2a653b4f8d3cd8e3fe99e0220176a4a7fc05fc58da404b636df5339c3ddd9869c6f31b5d2443723dfccdd4e9
e01210243532 83ff978ca37a35e6a0625666094c43a94810092bcbec98d82440f381e5efeffffffd5346d8c2d
27bc1c8585f755bc2f169af155601a044b16da689a4c8d05fe257b000000006b483045022100ed231c4a91
3916d2e69e640b239e88f8ccc9d0964f6e11041cfd74a6740ca5ec022052c3f975ef9ccf28e0a68876052e7
5b59be4d769f97fd5196e344b1ed1748938012102b8daf9e0a868aff8d9fc571c76164c244942030372d09
5c8058db0e33e67cf5bfeffffff0203b5ea0b000000001976a914a972ec745d1b162fe00eed8d7f4771188aa6
e9f188ac005ed0b2000000001976a914f161e0ad43db661c7a1dc012e5587f1eea2aef1888ac71000000"

图 2-26　hex 后的十六进制序列

3. **利用测试网络的 api 将上述 hex 序列广播到网络中去**。至此，一笔交易的广播已经实现完毕。

2.3.4.3 交易流程的界面实现

前文提到，我们为客户提供了实时查看交易进度的服务。在界面的实现方面，考虑到比特币交易过程的复杂性，我们将其做得简单易懂，如图 2-27 所示：

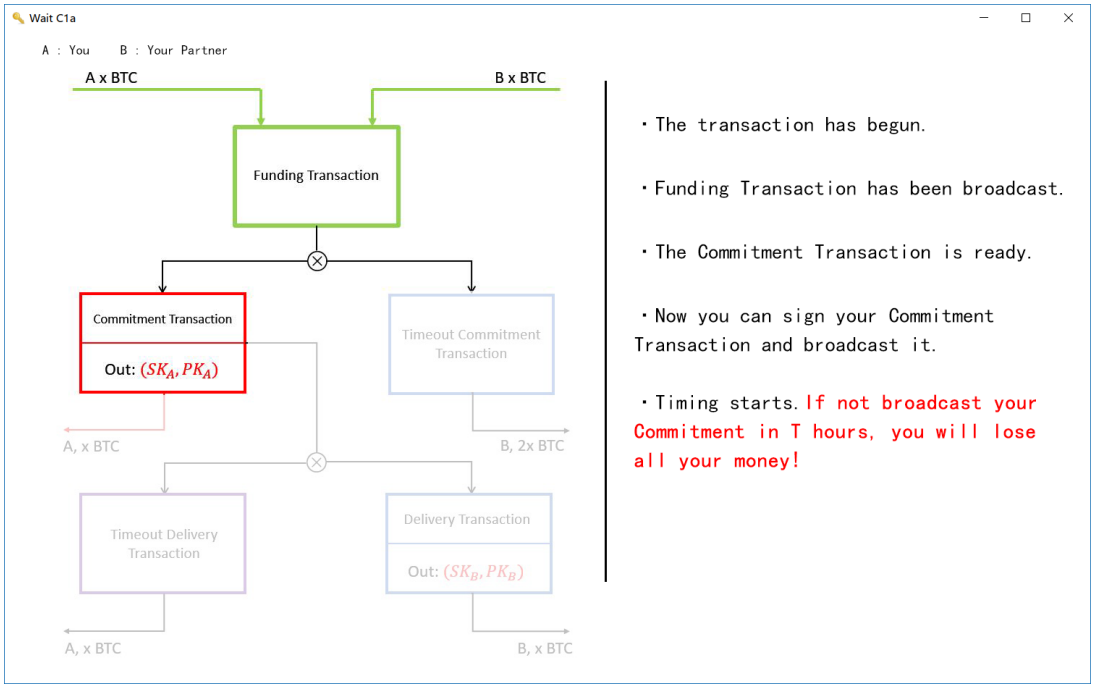

图 2-27　交易流程的界面展示

如此以来，只要用户对签名交换的方案稍有理解，便可以从界面中的流程图示与右侧的文字提示中获取目前的进度信息。

第三章 作品测试与分析

3.1 测试方案与参数设定

我们设定 A 和 B 使用“FairContract”对一份合同进行签署，A 是率先签名的一方，B 是滞后签名的一方。

设置双方商定好的保证金为 x=3 BTC，交易的赎回时间为 T=6 小时，这保证了合同的签署可以在半天左右的时间里完成。为了方便理解，我们在展示交易网络进程的界面中直接写入参数 x 和 T，而暂不代入具体值。

注意，比特币系统从广播交易到交易最终被确认是有一定时间的。为了提升测试效率，我们将系统接入 testnet 上进行测试，并且交易一旦广播，便手动使其正式入块，以此模拟矿工最终确认的结果，这样大大地加快了测试的进程。

3.2 测试环境

在 Windows10 系统下，客户端后台接入比特币的测试网络 testnet，调用 blockcypher 供测试的 api 来进行交易的广播。

3.3 测试过程：从双方的视角看一次合同签署完整过程的执行

3.3.1 合同到公私钥的等价转换：以 A 的视角

A 发起交易，B 接受。A 和 B 开始公私钥的生成阶段。

从 A 的视角看，如图 3-1 所示：

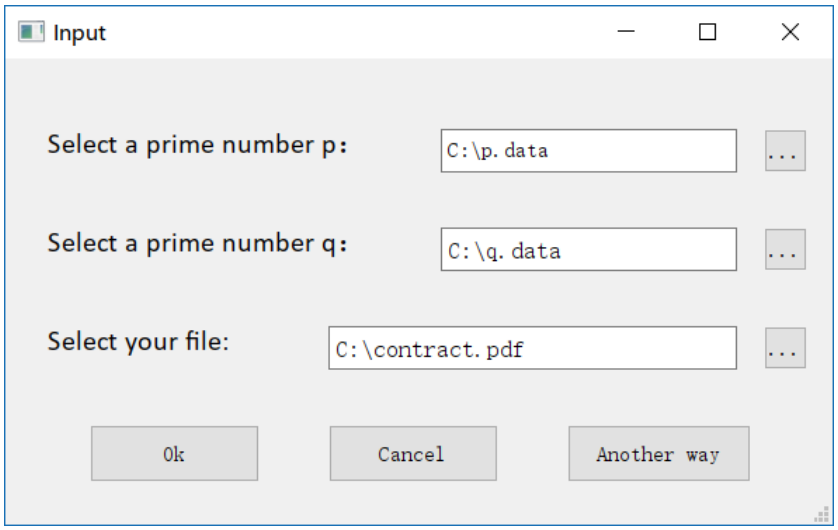

图 3-1　合同与参数 p，q 的上传界面

如果不信任 FairContract，可以通过其他途径协商，直接将输出的公钥输入系统即可。（若如此做，零知识证明过程由用户自行解决。）

B 视角也做了相同的事情。到此处，双方的与合同有关的公私钥都已产生，如图 3-2：

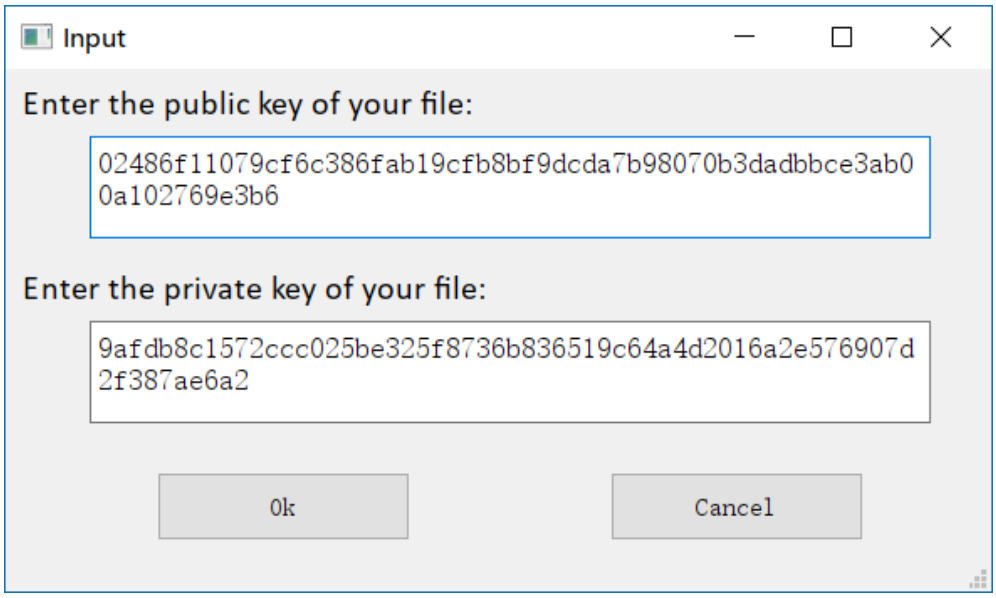

图 3-2　获取等价转换后的公私钥

公私钥以及相关的比特币地址已经存入本地数据库。

3.3.2 签名交换：A 的主要视角以及 B 的次要视角

3.3.2.1 双方对待定交易互签：以 A 的视角

以 A 的视角，对 B 构建的 Commitment Transaction 和 Delivery Transaction 进行签名（实际上是对 hex 序列的前向交易的 id 进行签名），如图 3-3 所示：

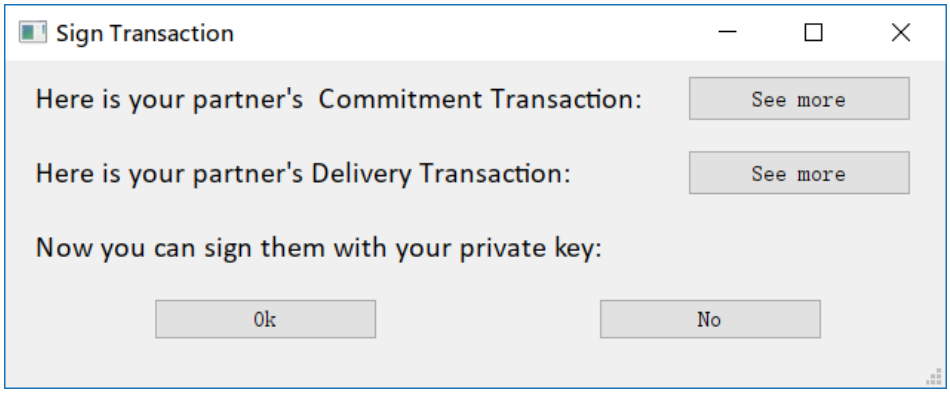

图 3-3　对交易单的签名窗口

3.3.2.2 正式交换过程：A 的主要视角以及 B 的次要视角

1. 还未进行交换，如图 3-4 所示：

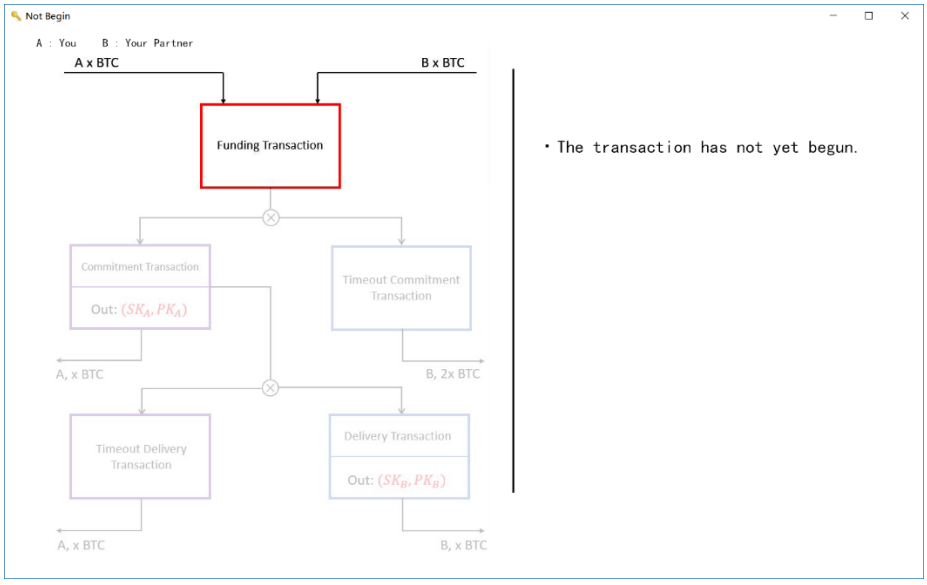

图 3-4　交易网络尚待开启

2. A 和 B 都对交易 F 进行了签名，F 被广播并入块，目前剩余时间小于 T 的设定值，系统督促 A 尽快签名并上传 C 交易，否则会丢失自己的所有比特币，如图 3-5 所示：

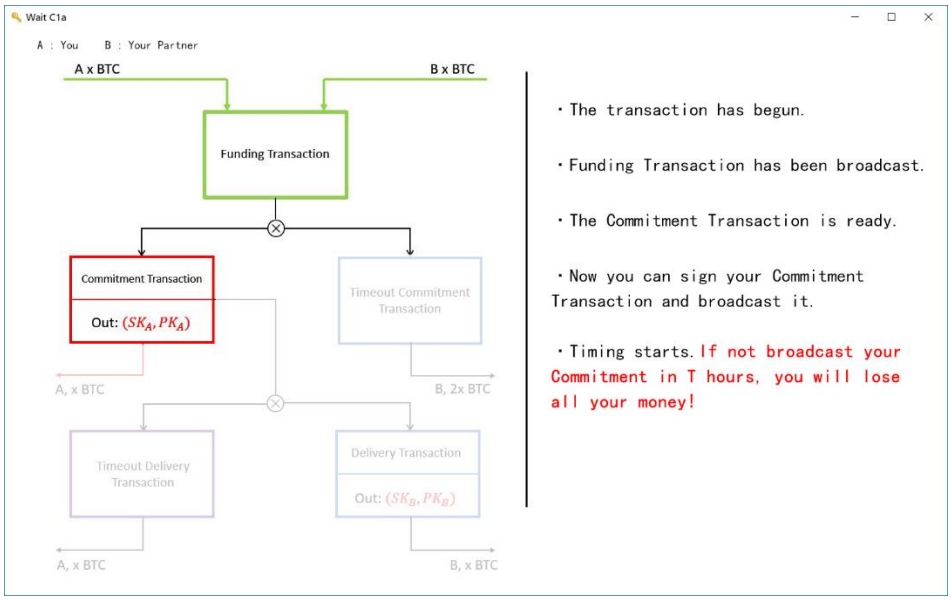

图 3-5　交易 F 已生效

3.A按时签名并发布了C交易，C交易入块后系统提示A已经拿回自己的比特币并且A的签名已经被B拿到，如图3-6所示；

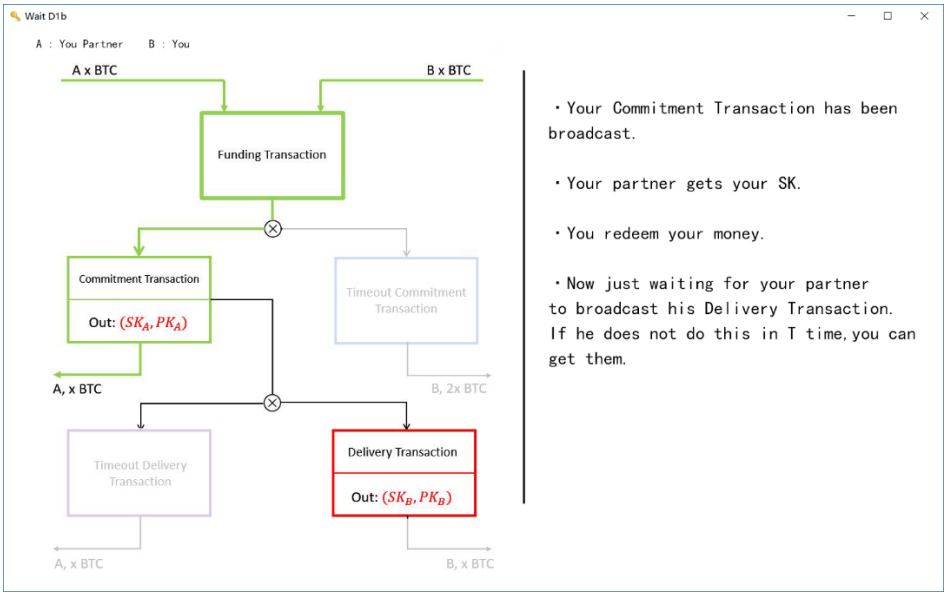

图 3-6　交易 C 已生效

4.B在设定时间T的限制内签名并发布了D交易，D交易入块后，系统提示A已经拿到B的签名，点击即可获取，如图3-7所示；

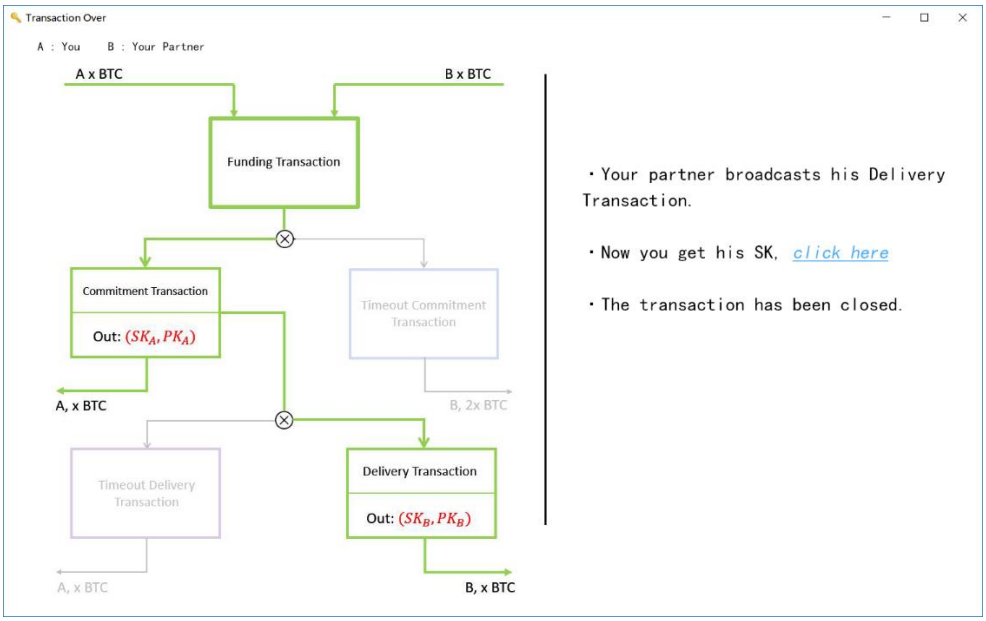

图 3-7　交易 D 已生效，A 已经拿到 B 关于合同的签名

至此，一次完整的签名交换过程结束。当然也存在双方失信的情况，我们分别在A的视角下对以下两种情况进行测试。

1. A没有在T时间内对C交易进行签名，系统会提示失信行为，并告知A其所有比特币会归属到B的账户下，如图3-8所示：

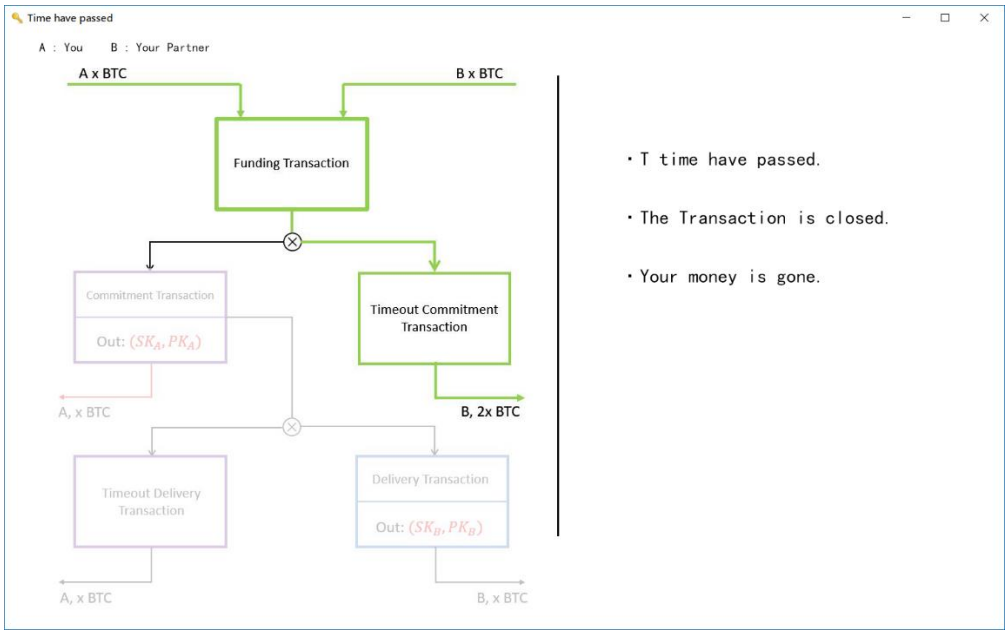

图 3-8　A 没有在 T 时间内对 C 交易进行签名，系统提示失信行为

至此，公平交换协议执行中止。

2.如果 B 没有在时间 T 内对 D 交易进行签名，系统提示 A 可以对 TD 交易进行签

名并广播，届时将得到 B 的所有比特币，同时无法获得 B 的签名，如图 3-9 所示：

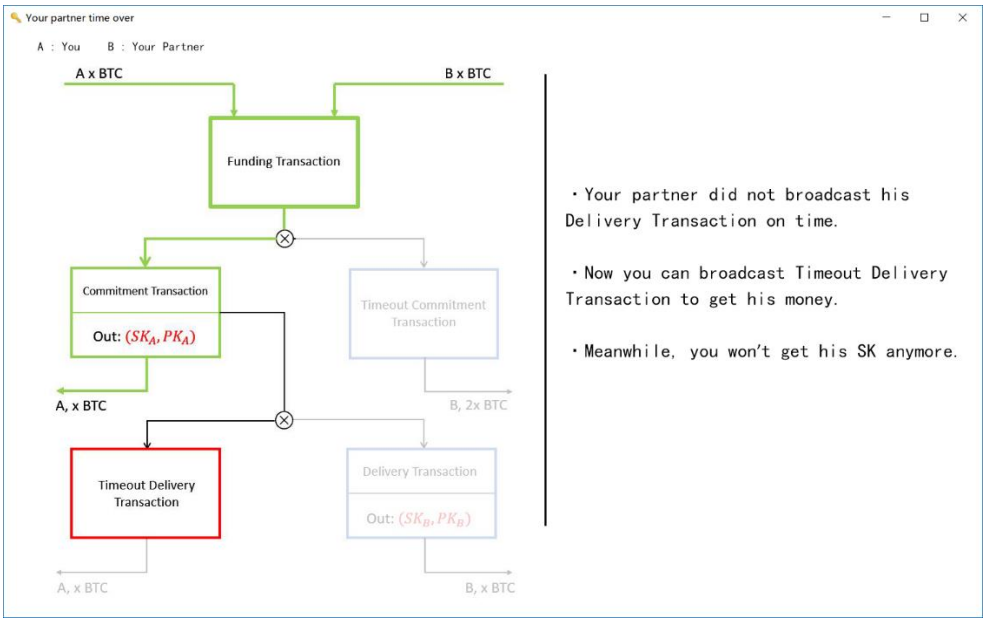

图 3-9　B 没有在时间 T 内对 D 交易进行签名，系统提示失信行为

至此，公平交换协议执行终止。

因为在签名交换的过程中 A 和 B 并非完全对称的关系，所以我们给出测试过程中三张 B 视角的截图。

1. A 发布 C 交易并入块后，系统会提示 B 在设定时间 T 内对 D 交易进行签名，否则就会失去自己的所有比特币，如图 3-10 所示：

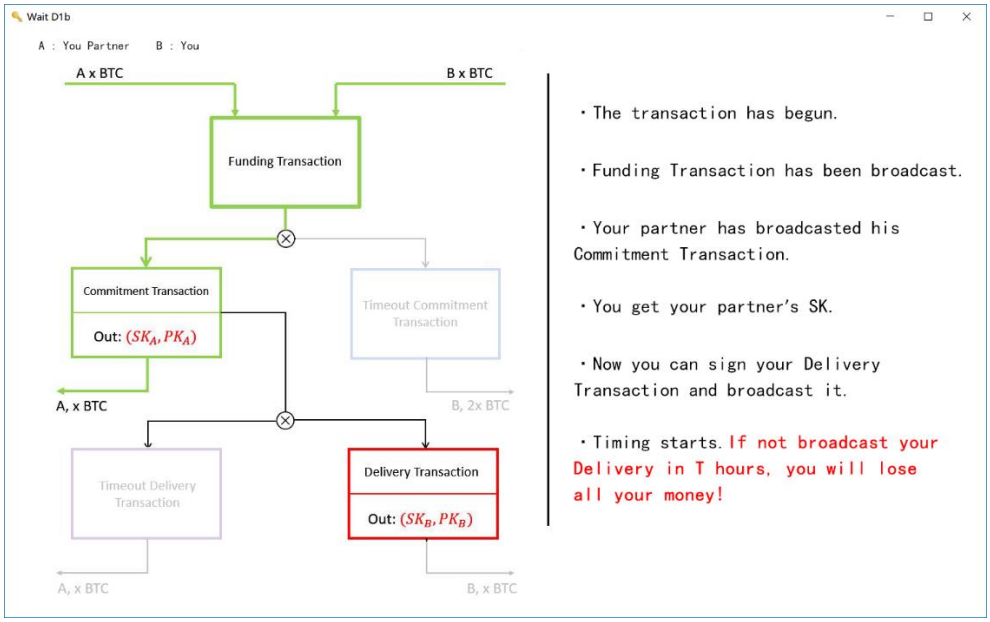

图 3-10　交易 C 生效

2. A,B 发布 F 交易后，若 A 没有在设定时间 T 内对 C 交易进行签名，系统会提示 B 可以签名并发布 TC 交易，拿走 A 的所有比特币，同时不能得到 A 的签名，如图 3-11 所示：

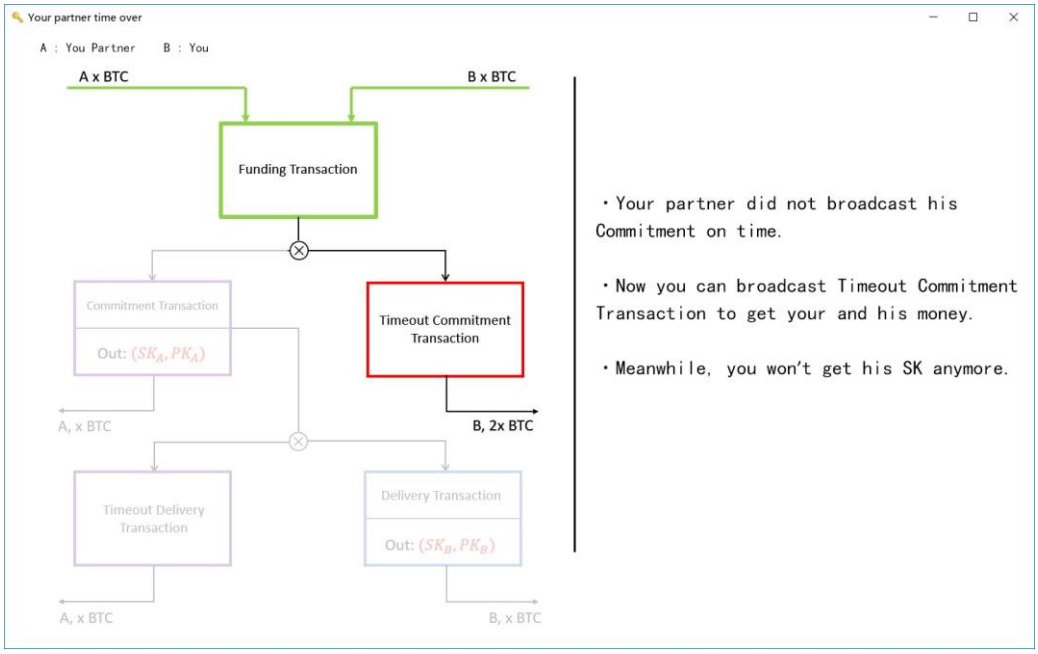

图 3-11　A 没有在设定时间 T 内对 C 交易进行签名，系统提示失信行为

至此，公平交换协议执行中止。

3. A 发布 C 交易后，若 B 没有在设定时间 T 内对 D 交易进行签名，则系统会告知 B 失信，会失去自己的所有比特币，如图 3-12 所示：

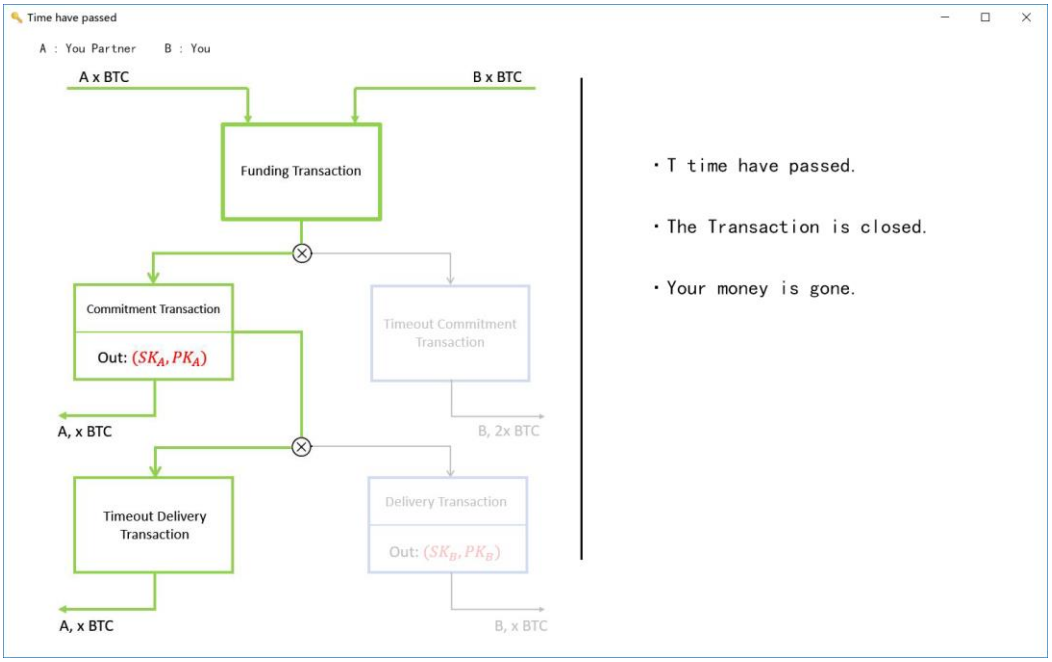

图 3-12　B 没有在设定时间 T 内对 D 交易进行签名，系统提示失信行为

至此，公平交换协议执行终止。

3.3 testnet 上账户余额变动查询：以 A 的视角

为了确认上述交易都成功入块，我们将系统接入 testnet，通过给出地址的方式查询余额，以验证交易的真实性。

在这里仅给出 A 视角的测试数据。

按照上文的设定，取保证金 x=3 BTC。

发布 F 交易之前，对 A 的账户进行查询，结果如图 3-13 所示：

```
>>> get_address_assets(myaddr1)

{u'address': u'mgN94Z7hRWy4UfZHTj2T8hZd1BctusLkps',
 u'balance': 332456390,
 u'final_balance': 332456390,
 u'final_n_tx': 27,
 u'n_tx': 27,
 u'total_received': 3321405100,
 u'total_sent': 948710,
 u'unconfirmed_balance': 0,
 u'unconfirmed_n_tx': 0}
```

图 3-13　F 交易尚未发布，A 的账户（图中的资产单位是 satoshi，1 BTC=10^8satoshi）

发布 F 交易之前，对 B 的账户进行查询，结果如图 3-14 所示：

```
>>> get_address_assets(myaddr2)

{u'address': u'mzPnKxCTMF3Dx9Nx9ktZ6n72LFheEX3MpQ',
 u'balance': 402110010,
 u'final_balance': 402110010,
 u'final_n_tx': 12,
 u'n_tx': 12,
 u'total_received': 402110010,
 u'total_sent': 0,
 u'unconfirmed_balance': 0,
 u'unconfirmed_n_tx': 0}
```

图 3-14　F 交易尚未发布，B 的账户（图中的资产单位是 satoshi，1 BTC=10^8satoshi）

F 交易生效之后，对 A 的账户进行查询，如图 3-15 所示,可以看出 A 的账户相比 F

交易发布之前（图 3-13）减少了 3 BTC.

```
>>> get_address_assets(myaddr1)

{u'address': u'mgN94Z7hRWy4UfZHTj2T8hZd1BctusLkps',
 u'balance': 32456390,
 u'final_balance': 32456390,
 u'final_n_tx': 28,
 u'n_tx': 28,
 u'total_received': 3321405100,
 u'total_sent': 300948710,
 u'unconfirmed_balance': 0,
 u'unconfirmed_n_tx': 1}
```

图 3-15　F 交易生效，A 的账户比图 3-13 少了 3 BTC（图中的资产单位是 satoshi，1 BTC=10^8satoshi）

F 交易生效之后，对 B 的账户进行查询，如图 3-16 所示,可以看出 B 的账户相比 F 交易发布之前（图 3-14）减少了 3 BTC.

```
>>> get_address_assets(myaddr2)

{u'address': u'mzPnKxCTMF3Dx9Nx9ktZ6n72LFheEX3MpQ',
 u'balance': 102110010,
 u'final_balance': 102110010,
 u'final_n_tx': 13,
 u'n_tx': 13,
 u'total_received': 402110010,
 u'total_sent': 300000000,
 u'unconfirmed_balance': 0,
 u'unconfirmed_n_tx': 0}
```

图 3-16　F 交易生效，B 的账户比图 3-14 少了 3 BTC（图中的资产单位是 satoshi，1 BTC=10^8satoshi）

C 交易生效之后，按照系统的设计，3 BTC 被发送到 A 的合同生成账户，为了方便测试，我们建立了一个新交易，通过 A 的合同生成账户将这笔资金支付给了 A 的原本账户（即本节所提的“A 的账户”）。对 A 的账户进行查询，如图 3-17 所示，可以看出 A 的账户相比 C 交易发布之前（图 3-15）增加了 3 BTC.

图 3-17　C 交易生效，A 的账户比图 3-16 增加了 3 BTC（图中的资产单位是 satoshi，1 BTC=10^8satoshi）

TD 交易生效之后，按照系统的设计，3 BTC 被发送到 A 的合同生成账户，为了方便测试，我们建立了一个新交易，通过 A 的合同生成账户将这笔资金支付给了 A 的原本账户（即本节所提的“A 的账户”）。对 A 的账户进行查询，如图 3-18 所示，可以看出 A 的账户相比 TD 交易发布之前（图 3-17）增加了 3 BTC.

图 3-18　TD 交易生效，A 的账户比图 3-17 增加了 3 BTC（图中的资产单位是 satoshi，1 BTC=10^8satoshi）

3.4 测试结果分析

从测试过程来看，FairContract 只具备提醒、计算、显示等功能，是一个纯粹的平

台，完全没有第三方的引导与干涉行为，且重要的步骤都由比特币系统来完成，真正地实现了去第三方；

另外，签名过程中 A 和 B 双方并非同步，都有彼此的“休息时间”，实现了异步签名，为双方客户留出了一定的空间。

从测试结果来看，图 3-13（含）到图 3-18（含）给出的账户资产变动符合系统的设计，说明整个过程中的交易都成功公布。

综上所述，FairContract 能够实现去中心化的合同签署过程。

第四章 创新性说明

4.1 完全去除第三方

不可否认第三方在互联网交换中一直是一个不可或缺的存在，然而第三方平台作为枢纽，一直是黑客攻击的重点，有着巨大的安全隐患。与此同时，无论是对于线上第三方还是离线第三方而言，构建一个能够信任的第三方平台也并非易事。也正是如此，大多数已有的第三方平台会收取价格不等的手续费，这对于交易双方而言又是一笔开支。

于是我们借助区块链技术，大胆的提出了这样一个完全去除第三方的方案，通过一系列的约束来确保交易的可行性与公平性，以此来达到去第三方的目的。

4.2 借助公认安全稳定的区块链平台

区块链是分布式数据存储、点对点传输、共识机制、加密算法等计算机技术的新型应用模式，是比特币的底层技术和基础架构。本质上是一个去中心化的数据库，同时作为比特币的底层技术。区块链是一串使用密码学方法相关联产生的数据块，每一个数据块中包含了一次比特币网络交易的信息，用于验证其信息的有效性（防伪）和生成下一个区块。

区块链具有去中心化，开放性，自治性，信息不可篡改的特点，其中去中心化特征是我们方案的核心。同时，区块链是目前公认的一个安全稳定平台，比特币已经在区块链上发展多年，其可靠性得以保证。我们在提出去第三方的目标后即想到借助去中心化的区块链平台，在保证了公认安全性的同时，为目标的实现节省了很多关于去第三方的算力，减少了设计与开发时间。

4.3 隔离见证技术的应用

我们在交易网络的构建上，参考了隔离见证技术。其实，每一个比特币交易，其实可以分为两部分。第一部分是说明结余的进出，第二部分是用来证明这个交易的合法性 (主要是签署)。第一部分可称为“交易状态”，第二部分就是所谓的“见证”。如

果只关心每个账户的结余，其实交易状态资料就已经足够。只有部分人(主要是矿工)才有必要取得交易见证。

中本聪设计比特币系统时，并没有把两部分资料分开处理，导致交易 ID 的计算混合了交易状态和见证。因为见证本身包括签署，而签署不可能对其自身进行签署，因此见证是可以由任何人在没有交易双方同意下可以改变的，造成所谓交易可变性。在交易发出后，确认前的交易 ID 可以被任意更改，因此基于未确认交易的交易是绝对不安全的。在 2014 年就曾有人利用这漏洞大规模攻击比特币网络，然而这问题一直迟迟得不到解决。

比特币核心开发 Pieter Wuille 在 2015 年 12 月于香港提出的隔离见证软分叉非常巧妙地彻底解决了这个问题。隔离见证用户在交易时，会把比特币传送到有别于传统的地址。当要使用这些比特币的时候，其签署 (即见证) 并不会记录为交易 ID 的一部份，而是另外处理。也就是说，交易 ID 完全是由交易状态 (即结余的进出) 决定，不受见证部份影响。

因此，基于以上原理，我们发现交易网络环节的设计是隔离见证的典型应用。通过隔离见证，可以让合同签署的双方在签名交换环节轻松地把握关键交易地进行，对自己的合同签名和保证金起到了保护作用。

4.4 一种新型的签名概念

本系统运用了一种新型的签名概念：将合同文本以及代表个人身份的信息等价转换为一个比特币账户的公私钥。这间接地将对合同签名等价转换为对输出脚本中包含由合同转化而来的公钥的交易单的签名。这原本是因为比特币交易单可修改内容的限制而做出的无奈之举，但惊喜的是，一方面我们给出了完备的安全验证，通过零知识证明的方式证明了其可行性；另一方面，我们看到了这种新型的签名概念的发展潜力。我们认为，这是数字签名与比特币系统的交汇，应用比特币系统来实现新型的签名，也将会是比特币系统接下来的发展趋势。

第五章 总结

公平交换、合同签署等这些在人们眼中非常简单的事情，因为第三方的存在，使得看似安全稳定的系统有被实施中心攻击的巨大隐患。如何将第三方的角色完全剔除，是密码学家一直在探讨的问题。本文所介绍的FairContract PC客户端——一种公平的异步合同签署系统基于“无信任”的原则，大胆构建出一种新型的签名等价转换的方案，通过零知识证明以及和比特币系统的连接，运用隔离见证技术，实现了去第三方的合同签署过程。

在第一章中，本文简要介绍了当前有第三方协议的概况以及比特币系统发展所带动的潮流，构建了一个简单的“不信任”型的场景，提出了FairContract客户端系统的构建，并对系统的内容进行了综合叙述。随后在第二章，本文详细介绍了FairContract的系统功能、结构设计与软件实现，包括合同签名的等价转换与零知识证明，隔离见证交易的搭建与实施过程，以及各个子交易在脚本语言下的实现。第三章中，本文给出了FairContract在各个环节的测试截图，整个合同签署过程执行按部就班，有条不紊，最终比特币testnet返回的账户余额变动符合预期。第四章，本文分析了此系统的四处创新点：完全去除第三方、借助公认安全稳定的区块链平台、隔离见证技术的应用以及一种新型的签名概念。这些都是此系统的突出之处。

无论比特币系统如何发展，交换协议如何被优化，安全永远是亘古不变的主题，是一个系统的核心指标。我们将持续关注公平交换、数字签名等问题的最新进展，为签名交换问题乃至多方计算、安全证明等研究热点提出更好的解决方案！

参考文献

[1]刘景伟. 电子商务中的公平交换协议研究[D].西安电子科技大学,2007.

[2] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.

[3] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments[J]. 2015.

[4] Okupski K. Bitcoin Developer Reference[J]. http://enetium. com/resources/Bitcoin. pdf, 2014.

[5] BlockCypher’s API. https://www.blockcypher.com/dev/bitcoin.