Version 1.0.0

万维链黄皮书

目录

1 密码学基础知识

1.1 椭圆曲线基础知识

1.1.1 椭圆曲线简介

设G表示一个有限域，在其上定义一个椭圆曲线E，实际上这个曲线E表示为一个点的集合，则有

$$E/G = \{(x,y)|y^2 + a_1xy + a_3y = x^3 + a_2x_2 + a_4x + a_6,$$
$$a_1, a_3, a_2, a_4, a_6, x, y \in G\} \cup \{O\}$$

其中，O表示无穷远点。

在椭圆曲线E上定义加法运算，P和Q是椭圆曲线E上的两个点，则对于这两个点的加法运算有$P + Q = R$。这里R表示为过点P和Q的直线与曲线E的交点关于x轴对称的椭圆曲线上的点。此时如果当$P = Q$时，则R表示为P点的切线与曲线E的交点关于x轴对称的椭圆曲线上的点。这样，在有限域G上$(E, +)$则构成了阿贝尔群，且其加法单位元为O。

设$P = (x_1, y_1) \in E, Q = (x_2, y_2) \in E$，如果$x_1 = x_2$且$y_1 = y_2$，那么则有$P + Q = O$；否则，加法运算$P + Q = (x_3, y_3)$，这里的 $x_3 = \lambda^2 - x_1 - x_2, y_3 = \lambda(x_1 - x_3) - y_1$，其中

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, P \neq Q \\ \dfrac{3{x_1}^2 + a}{2y_1}, P = Q \end{cases}$$

1.1.2 ECDSA 签名算法

定义一个椭圆曲线$E_p(a, b)$和其上的基点P，其中N是P的阶。Q为曲线$E_p(a, b)$上任意一点，令$Q = kP$建立公私密钥对，其中k为私钥，Q为公钥可以公开。

下面给出发送者 A 和接收者 B 之间利用椭圆曲线密码进行数字签名的具体过程：

Step1：首先利用 Hash 函数对明文消息M进行计算，其中，常用的 Hash 函数算法有 MD5 算法或 SHA-1 算法，可以计算出明文消息M的摘要值$t = H(M)$;

Step2：然后在区间$[1, N - 1]$范围内随机选取一个整数k作为此次签名的私钥;

Step3：计算出公钥$Q = kP$;

Step4：计算$r = Q_x \ mod \ N$，其中，Q_x是表示公钥Q的横坐标，如果$r = 0$，则返回到 Step2;

Step5：计算$s = k^{-1}(t + rk) \ mod \ N$，其中$k$为发送者 A 的私钥，如果$s = 0$，则返回到 Step2;

Step6：发送者 A 把消息签名(r, s)传送给接收者 B。

接收者 B 收到消息签名(r, s)后，对消息签名的具体验证过程如下：

Step1：首先对消息签名r和s进行验证，即判断其是否是在区间$[1, \mathrm{N} - 1]$范围内的正整数，如果该签名不符合消息签名的条件，则认为收到的消息签名(r, s)不是有效合法的签名;

Step2：根据所获得发送者 A 的签名公钥Q，利用发送者 A 和接收者 B 具有相同的 Hash 函数摘要值，计算出待签名明文消息M的摘要值$t = H(M)$;

Step3：计算出参数值$e = s^{-1} \ mod \ N$;

Step4：计算出参数值$u = te \ mod \ N$;

Step5：计算出参数值$v = re \ mod \ N$;

Step6：计算出参数值$R = uP + vQ$;

Step7：如果$R = 0$，则接收者 B 可以拒绝签名。否则，计算$r' = R_x \ mod \ N$，其中R_x是表示参数R的横坐标;

Step8：如果所计算的参数值r'与r是相同的，则可以认为发送者 A 对明文消息M的签名被接收者 B 验证通过，即该签名是合法有效的。否则，该签名不是合法有效的，接收者 B 可以拒绝此

签名。

基于椭圆曲线密码算法的数字签名方法，一方面是因为这种方案能够避免求阶运算中的模逆运算，因而比基于离散对数方案的签名算法要简单;另一方面则是因为计算明文消息摘要$H(M)$要比计算$H(M,R)$简单，所以其运算速度要比 Schnorr 数字签名方案的运算速度快得多。因此，基于椭圆曲线密码的数字签名方案在抗攻击的安全强度、密钥长度、运算速度、计算代价与带宽要求等方面中具有很好的应用优势。

1.2 门限密钥共享技术

1.2.1 Shamir 门限密钥共享概念

门限密钥共享技术（threshold key sharing scheme）解决的是密钥安全管理问题。现代密码学体制的设计是使得密码体制的安全性取决于密钥的安全，密钥的泄露就意味着体制失去了安全性，因此密钥管理在密码体制的安全性研究和设计中占有重要的地位。特别是多方利益体共同管理一个账户时，账户的密钥如何可信安全地分配给多方参与者就变得非常棘手。针对这一问题，以色列密码学家 Shamir 提出了 Shamir(k,n)门限密钥共享的概念：密钥被分为n份分配给n个参与者，每个参与者掌握一个密钥份额（key sharing），只有集齐超过k个密钥份额，才能够将密钥恢复。

1.2.2 线性密钥共享机制

线性密钥共享是 Shamir 门限密钥共享的推广，它的本质是要求主密钥空间、子密钥空间和随机输入集合都是线性空间，并且密钥重构函数是线性的。形式化定义如下：
设K是一个有限域，Π是实现存取结构AS的一个密钥共享体制，$S \subset K$是主密钥空间。我们说Π是K上的一个线性密钥共享体制，如果满足以下条件：

1) 子密钥是K上的线性空间，即对于$\forall i$，$\exists$常数d_i，使得子密钥空间$S_i \subset K^{d_i}$。记$\Pi_{i,j}(s,r)$为P_i收到空间K^{d_i}的向量的第j个分量，这个分量依赖于主密钥s和随机数r
2) 每个授权集都可以通过线性组合子密钥的方式得到主密钥，即对于任意一个授

权集$G \in AS$，∃常数$\{a_{i,j}: P_i \in G, 1 \le j \le d_i\}$，使得对任意主密钥$s$和随机数$r$，都有$s = \sum_{P_i \in G} \sum_{1 \le j \le d_i} a_{i,j} \cdot \Pi_{i,j}(s, r)$

1.2.3 Shamir 多项式插值门限密钥共享方案

Shamir 结合有限域上多项式的特点和拉格朗日重构多项式理论，设计了基于拉格朗日插值多项式的门限密钥管理方案，方案如下：

- 密钥分享过程

1) 密钥分享者要分享密钥s，构造有限域上随机的$k-1$次多项式：$f(x) = s + a_1 x + \cdots + a_{k-1} x^{k-1}$，显然$f(0) = s$

2) 密钥分享者随机选取随机数$x_1, x_2 \ldots, x_n$，并计算出$f(x_1), f(x_2) \ldots, f(x_n)$

3) 每个参与者P_i获得的密钥份额为$s_i = (x_i, f(x_i))$

- 密钥还原过程：

根据生成密钥份额算法可知点s_i在曲线$f(x)$上，根据拉格朗日内插法重构多项式可知，任意k个点可以重构出$k-1$次多项式，即有：

$$f(x) = \sum_{i=1}^{k} f(x_i) \prod_{j=1, j \ne i}^{k} \frac{x - x_j}{x_j - x_i}$$

另上面多项式$x = 0$，即可还原出密钥s：

$$s = \sum_{i=1}^{k} f(x_i) \prod_{j=1, j \ne i}^{k} \frac{x_j}{x_j - x_i}$$

可简写如下：

$$s = \sum_{i=1}^{k} \mathrm{b}_i f(x_i)$$

其中

$$b_i = \prod_{j=1, j\neq i}^{k} \frac{x_j}{x_j - x_i}$$

1.3 安全多方计算

1.3.1 安全多方计算提出背景

随着互联网的迅速发展，越来越多的应用场景需要网络用户之间进行协作运算。但是出于隐私保护和数据安全的考虑，参与协作运算的用户并不想和其他用户进行计算数据的分享，这一问题导致协同计算无法执行，从而导致网络资源并不能够高效分享利用和一些应用场景难以实现。安全多方计算（secure multi-party computation）让这一问题迎刃而解，它为解决数据隐私保护和协同计算之间的矛盾提供了理论基础。

安全多方计算是分布式密码学的理论基础，也是分布式计算研究的一个基本问题。安全多方计算是指在一个互不相信的多用户网络中，两个或多个用户能够不泄漏各自私有输入信息，协同合作执行某项计算任务。简单地说，安全多方计算是指一组人，比如P_1……P_n，共同安全地计算函数$f(x_1, \dots\dots, x_n) = (y_1, \dots\dots, y_n)$。其中这个函数的$n$个输入分别由这$n$个参与者秘密掌握的，设$P_i$的秘密输入是$x_i$，并且在计算结束后，$P_i$得到输出$y_i$。这里的安全性是要求即使在某些参与者有欺骗行为的情况下，保证计算结果的正确性，即计算结束后每个诚实的参与者P_i都能得到正确的输出y_i，同时还要求保证每个参与者输入的保密性，即每个参与者P_i除了(x_i, y_i)外，得不到任何其他信息。安全多方计算已经有了丰富的理论成果和强有力的工具，虽然它的实际应用还处于起步阶段，但终将成为计算机安全一个不可缺少的部分。

1.3.2 安全多方计算协议的分类

目前安全多方计算协议根据实现方式的不同，可以分为四类：

- 基于 VSS 子协议的安全多方计算协议

现存的大部分安全多方计算协议都采用了可验证密钥分享 VSS（Verifiable Secret Sharing）子协议作为协议构造的基础，这类协议适合计算任意有限域上的函数。有限域上任意函数均可表示成域中定义加法和乘法的有向图，因此只要可以安全计算加法和乘法，就可以计算每一个加法和乘法来完成有限域上任意函数的计算。

- 基于 Mix-Match 的安全多方计算协议

基于 VSS 子协议的安全多方计算协议能够计算任意函数，但是不能高效计算布尔函数，为此提出了另一种安全多方协议——Mix-Match。这种协议的基本思路是参与者使用秘密分享方案分享系统的私钥，系统的公钥公开。协议过程中，参与者将自己输入使用的公钥 y 随机加密，然后将自己的加密结果公布，最后通过 Mix-Match 使得所有参与者获得共同输出。

- 基于 OT 的安全多方计算协议

基于 OT 的安全多方计算协议用于计算任意比特运算函数。它利用 OT 子协议实现“与（and）”、“或（or）”、“非（not）”三种基础运算，再将任意比特运算函数分解为三种基础运算的组合，最后通过迭代的方式计算出任意比特运算函数。

- 基于同态加密的安全多方计算

基于同态加密的安全多方计算能够抵抗主动攻击，它的思路是选定原子计算，使得任意函数计算均可分解为原子计算的序列，同时原子计算的输入和输出均使用同态加密算法加密，在加密状态下得到最终的计算结果，只有特定的参与者集合才能够将计算结果解密得到明文。

1.4 环签名简介

2001 年，Rivest 等人在如何匿名泄露秘密的背景下提出了一种新型签名技术，称为环签名（Ring Signature）。环签名可以被视为一种特殊的群签名（Group Signature），由于群签名需要可信中心和安全的建立过程，往往在匿名保护上存在漏洞（签名者对于可信中心是可追溯的），而环签名在群签名基础上去除了可信中心和安全建立过程，对于验证者来说，签名者是完全匿名的，所以环签名更具实用价值。

自环签名提出后，大量学者发现其重要的价值，基于椭圆曲线、门限等多种环签名被大量设计开发，总体概括可分为四类：

1. 门限环签名
2. 关联环签名
3. 可撤销匿名性环签名
4. 可否认环签名

为实现区块链上智能合约的代币交易隐私性，我们使用一种关联环签名，以实现隐私性的同时防止双花问题。

2 基于安全多方计算和门限密钥共享的锁定账户生成方案

2.1 安全多方计算的基础运算

加法、乘法、一元求逆运算为有限域上的三种基础运算，任意计算都可以分解为这个有限域上加法、乘法、一元求逆运算的序列，因此只要能够完成三种基础运算的多方计算，那么有限域上任意计算过程均可通过基础运算的多方计算协议去迭代得到。以下将介绍在基于拉格朗日插值多项式的秘密分享方案下，有限域上基础运算的安全多方计算算法。

2.1.1 加法

在基于拉格朗日插值多项式的秘密分享方案下，需要确定一个多项式，分享的秘密是这个多项式的常数项，而秘密份额则为这个多项式在某点处的取值。不妨设α、β为两个分享的秘密，对应的多项式为$f_\alpha(x)$、$f_\beta(x)$，参与者P_i拥有的秘密份额分别为$\alpha_i = f_\alpha(i)$、$\beta_i = f_\beta(i)$。参与者P_i要得到秘密$\alpha+\beta$的秘密份额，就需要构造一个多项式$f(x)$，使得这个多项式的常数项为$\alpha+\beta$，且P_i能够计算得到$f(i)$。构造过程如下：

$\because$ α_i、β_i为秘密α、β的秘密份额，且对应多项式为$f_\alpha(x)$、$f_\beta(x)$

$\therefore$ $f_\alpha(x) = \alpha + a_{\alpha,1}x + \cdots + a_{\alpha,k-1}x^{k-1}$ $f_\beta(x) = \beta + a_{\beta,1}x + \cdots + a_{\beta,k-1}x^{k-1}$

不妨定义 $f(x) = f_\alpha(x) + f_\beta(x)$，则 $f(i) = f_\alpha(i) + f_\beta(i) = \alpha_i + \beta_i$

显然$f(x)$为$k-1$次多项式，且常数项为$\alpha+\beta$，$f(i)$为这个多项式在 i处取值

$\therefore\ \gamma_i = f(i)$即为秘密$\alpha + \beta$的秘密份额

由以上构造过程得到加法的安全多方计算算法：

加法的多方计算算法

输入：秘密α、β的秘密份额α_i、β_i

输出：秘密$\alpha + \beta$的秘密份额γ_i

1) $\gamma_i = \alpha_i + \beta_i$

2.1.2 乘法

设α、β为两个分享的秘密，对应的多项式为$f_\alpha(x)$、$f_\beta(x)$，参与者P_i拥有的秘密份额分别为$\alpha_i = f_\alpha(i)$、$\beta_i = f_\beta(i)$。如果参与者直接在本地计算秘密份额α_i、β_i的乘积，虽然计算之后分享秘密$\alpha\beta$的多项式常数项确实为$\alpha\beta$，但是多项式的次数为$2(k-1)$，因此需要降低多项式的次数。

$f_\alpha(i)$、$f_\beta(i)$为参与者P_i拥有的秘密份额，$f_\alpha(x)$和$f_\beta(x)$乘积为：

$$f_{\alpha\beta}(x) = f_\alpha(x) f_\beta(x) = \alpha\beta + a_1 x + \cdots + a_{2(k-1)} x^{2(k-1)}$$

$$f_{\alpha\beta}(i) = f_\alpha(i) f_\beta(i)\ ,\ 1 \le i \le 2(k-1) + 1$$

用矩阵表示为：

$$\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & (2k-1)^{2(k-1)} \end{bmatrix} \begin{bmatrix} \alpha\beta \\ \vdots \\ a_{2(k-1)} \end{bmatrix} = \begin{bmatrix} f_{\alpha\beta}(1) \\ \vdots \\ f_{\alpha\beta}(2k-1) \end{bmatrix}$$

将上面系数矩阵记做A，则显然A是一个非奇异矩阵，A的逆矩阵记为A^{-1}，它是一个常

数矩阵。记$(\lambda_1,\cdots,\lambda_{2k-1})$是矩阵$A^{-1}$的第一行，则有：

$$\alpha\beta = \lambda_1 f_{\alpha\beta}(1) + \cdots + \lambda_{2k-1} f_{\alpha\beta}(2k-1)$$

每个参与者随机选择$2k-1$个$k-1$次多项式$h_1(x),\cdots,h_{2k-1}(x)$，要求满足$h_i(0) = f_{\alpha\beta}(i)$。

定义$H(x) = \sum_{i=1}^{2k-1} \lambda_i h_i(x)$，显然有：

$$H(0) = \sum_{i=1}^{2k-1} \lambda_i h_i(0) = \lambda_1 f_{\alpha\beta}(1) + \cdots + \lambda_{2k-1} f_{\alpha\beta}(2k-1) = \alpha\beta$$

$$H(j) = \sum_{i=1}^{2k-1} \lambda_i h_i(j)$$

因此$H(x)$即为分享秘密$\alpha\beta$的多项式，且$H(i)$为秘密份额。

乘法的多方计算算法

输入：秘密α、β的秘密份额α_i、β_i

输出：秘密$\alpha\beta$的秘密份额γ_i

1) P_i产生一个随机的$k-1$次多项式$h_i(x)$，要求满足$h_i(0) = \alpha_i\beta_i$

2) P_i计算$h_i(j)$发送给P_j，$1 \le j \le 2k-1$

3) 每个参与者P_i收集其他参与者发送给它秘密份额，最后计算$\gamma_j = H(i) = \sum_{j=1}^{2k-1} \lambda_j h_j(i)$，$H(i)$即为 P_i获得的秘密$\alpha\beta$的秘密份额

2.1.3 一元求逆

设α为分享的秘密，对应的多项式为$f_\alpha(x)$，参与者P_i拥有的秘密份额为$\alpha_i = f_\alpha(i)$。一元

求逆运算是指参与者P_i由秘密份额α_i计算得到α^{-1}的秘密份额$f_{\alpha^{-1}}(i)$，而且在计算过程中不能泄露α、α^{-1}以及二者的秘密份额。计算思路如下：

参与者P_i选择随机数r_i，并选择随机多项式$g_i(x)$计算其秘密份额$r_{ij} = g_i(j)$发送给参与者P_j。接受到所有秘密份额之后，P_j计算$r_j' = \sum_{l=1}^{n} r_{l,j}$。这样所有参与分享了同一个随机数$r = r_1 + \cdots + r_n$。再利用乘法的多方计算算法，通过$\alpha$和$r$的秘密份额计算得到$\alpha r$的秘密份额$f_{\alpha r}(i)$，并发送给其他参与者，因此可以利用拉格朗日插值法恢复出αr，不妨设$m = \alpha r$。显然令$f_{\alpha^{-1}}(i) = m^{-1} r_i'$，即为$\alpha^{-1}$的秘密份额。

一元求逆的多方计算算法

输入：秘密α的秘密份额α_i

输出：秘密α^{-1}的秘密份额γ_i

1) P_i选择随机数r_i，并选择随机多项式$g_i(x)$计算其秘密份额$r_{ij} = g_i(j)$发送给参与者P_j，$1 \le j \le n$

2) P_j接受到所有秘密份额之后，计算$r_j' = \sum_{l=1}^{n} r_{l,j}$

3) 利用乘法的多方计算算法，通过α和r的秘密份额计算得到αr的秘密份额$f_{\alpha r}(i)$，并恢复出αr

4) 令$m = \alpha r$，$\gamma_i = f_{\alpha^{-1}}(i) = m^{-1} r_i'$，即为$P_i$掌握的$\alpha^{-1}$秘密份额。

2.2 锁定账户生成方案

锁定账户生成方案是基于拉格朗日插值多项式的门限密钥管理方案的改进。它的基本思路是通过门限密钥分享，所有验证节点以去中心化的方式共同生成锁定账户，并且每一个验证节点都掌握着锁定账户私钥的一个份额。这样保证了锁定账户私钥是以私钥份额这种分布式的形式存在于整个网络中，因此才能够去中心化的管理。 具体算法如下：

(k,n)门限锁定账户生成算法

1) P_i选择随机数d_i，将d_iG广播全网（G为椭圆曲线基点）

2) P_i选择$k-1$次多项式：$f_i(x)=d_i+a_{i,1}x+\cdots+a_{i,k-1}x^{k-1}$，将$f_i(j)$通过安全信道传输给$P_j$，同时将$a_{i,1}G$……$a_{i,k-1}G$广播全网

3) P_j到P_i信息之后，进行验证：$\sum_{t=0}^{k-1} j^t\, a_{i,t}G=f_i(j)G$，如果验证不通过，则拒收，请求$P_i$重新发送信息

4) 待所有信息都发送完毕且验证通过后，每个用户的密钥份额为$t_s=\sum_{j=1}^{n} f_j(s)$，$s=1,\ldots\ldots,n$

5) (k,n)门限锁定账户地址为$address=Hash(Q)$，其中$Q=\sum_{i=0}^{d} d_i\,G$，

对应私钥为$privatekey=\sum_{i=0}^{d} d_i$，这个私钥只有通过$k$份以上密钥份额才能够恢复

2.3 锁定账户签名方案

锁定账户签名算法采用 ECDSA 签名算法，因为它是目前区块链项目的主流签名算法，这个选择能够提高系统的兼容性。在锁定账户签名生成过程中，不同于原始 ECDSA 签名算法，账户私钥和随机数是以多方计算的形式参与到 ECDSA 签名过程中；锁定账户签名验证过程与原始 ECDSA 签名验证算法相同。因此只介绍锁定账户签名生成过程，具体如下：

锁定账户签名生成算法

1) 参与节点利用多方计算共享随机数c，P_i的随机数份额为c_i

2) P_i计算$R_i = c_i G$，并广播R_i

3) 各节点广播结束后，P_i计算$(x, y) = \sum_{j=1}^{k} b_j R_j$，$r = x \bmod p$，其中$b_j = \sum \frac{j}{j-i}$

4) 参与节点P_i利用一元求逆多方计算算法计算c^{-1}的份额ω_i

5) 利用ω_i和t_i，通过乘法多方计算算法，计算得到$c^{-1}d$的份额v_i，其中d为锁定账户私钥，t_i为锁定账户私钥份额

6) 计算结束后，P_i计算$s_i = \omega_i m + v_i r$，s_i即为参与者P_i的签名份额，P_i将其广播

7) P_i计算验证$R_j = u_{j1} G + u_{j2} Q_j$，其中$u_{j1} = m s_j^{-1}$，$u_{j1} = r s_j^{-1}$，$Q_j = t_j G$， 如果验证通过，则接受签名份额$s_j$；否则拒绝签名份额$s_j$

8) P_i 接受到k个以上签名份额后，利用拉格朗日插值算法还原出完整签名s，最终签名为(r, s)

2.4 锁定账户密钥更新方案

本方案中采用的基于拉格朗日插值多项式的门限密钥共享算法属于线性密钥共享机制，因此密钥共享满足同态性：密钥key_1的(k, n)门限密钥份额为$(a_1, \cdots, a_n)$，密钥key_2的(k, n)门限密钥份额为$(b_1, \cdots, b_n)$，则$(a_1 + b_1, \cdots, a_n + b_n)$为密钥$key_1 + key_2$的$(k, n)$门限密钥份额。如果令$key_2 = 0$，那么我们就能够得到$key_1$的新的$(k, n)$门限密钥份额。具体算法如下：

锁定账户密钥更新算法

1) 节点P_i选择随机多项式，将 0 进行共享，并计算份额$(f_i(1),\cdots,f_i(n))$

2) 节点P_i通过安全信道将$f_i(j)$发送给P_j，$j=1,\ldots\ldots,n$

3) 所有发送结束后，节点P_i收到信息$(f_1(i),\cdots,f_n(i))$，则P_i新的密钥份额为：$t_i^{new}=t_i+\sum_{j=1}^{n}f_j(i)$

3 交易隐私保护方案

3.1 环签名方案

环签名可分为四个部分：GEN，SIG，VER，LNK。

GEN：采集公共参数，随机选取$n-1$个公钥，合同用户公钥P构成公钥集合$S=\{P_i|i=1,2\ldots,n\}$；对用户公私钥对(P,x)，$x\in[1,l-1]$，l为点P的阶，生成公钥镜像I。

SIG:针对所需签名消息m,利用公钥集合$S=\{P_i|i=1,2\ldots,n\}$,其中P_s为用户真实公钥，计算输出签名$ringsig$。

VER:基于消息m,公钥集合S和签名$ringsig$,验证签名合法性,输出“True”或“False”。

LNK：利用集合$\mathcal{I}=\{I_i\}$，判断$ringsig$签名是否已使用。

具体过程介绍如下：

GEN:签名者使用公私钥对(P,x),有$P=xG$,计算$I=xH_p(P)$,其中H_p为 hash 函数，输出椭圆曲线上一个随机的点。随机选取$n-1$个公钥合同P构成公钥地址集$S=\{P_i|i=1,2\ldots,n\}$，其中$P_s=P$。

SIG：签名者选择随机数$\{q_i|i=1,2,\ldots,n\}$和$\{\omega_i|i=1,2,\ldots,n,i\neq s\}$，做如下计算：

$$L_i=\begin{cases}q_iG, & if\ \ i=s\\ q_iG+\omega_iP_i, & if\ \ i\neq s\end{cases}$$

$$R_i=\begin{cases}q_iH_p(P_i), & if\ \ i=s\\ q_iH_p(P_i)+\omega_iI, & if\ \ i\neq s\end{cases}$$

计算：

$$c = H_s(m, L_1, \dots, L_n, R_1, \dots, R_n)$$，H_s为 hash 函数

计算：

$$c_i = \begin{cases} \omega_i, & if\ \ i \neq s \\ c - \sum_{i=1, i\neq s}^{n} c_i \ \ mod\ l, & if\ \ i = s \end{cases}$$

$$r_i = \begin{cases} q_i, & if\ \ i \neq s \\ q_s - c_s x \ \ mod\ l, & if\ \ i = s \end{cases}$$

最后生成签名：

$$ringsig = (I, c_1, \dots, c_n, r_1, \dots, r_n)$$

VER：验证者验证签名时，基于消息m、公共参数和$S = \{ P_i | i = 1,2 \dots, n \}$、$ringsig = (I, c_1, \dots, c_n, r_1, \dots, r_n)$，计算：

$$\begin{cases} L_i' = r_i G + c_i P_i, \\ R_i' = r_i H_p(P_i) + c_i I \end{cases}$$

然后验证等式$\sum_{i=1}^{n} c_i = H_s(m, L_1', \dots, L_n', R_1', \dots, R_n') \ mod\ l$是否成立，若成立，则签名有效，执行 LNK。

说明：验证过程发现等式成立即需要$L_i' = L_i$，$R_i' = R_i$，以L_i'为例说明如下：

$$L_i = \begin{cases} q_i G, & if\ \ i = s \\ q_i G + \omega_i P_i, & if\ \ i \neq s \end{cases}$$

$$L_i' = r_i G + c_i P_i,$$

$i \neq s$时，有$q_i = r_i$，$\omega_i = c_i$，显然成立，而$i = s$时：$L_s = q_s G = (r_s + c_s x)G = r_s G + c_s P_s = L_s'$，综上有$L_i' = L_i$，同理$R_i' = R_i$成立，则说明 VER 过程是正确的。

LNK：对于区块链上已经出现过的所有I构建集合$\mathcal{I}$，若当前I在集合中出现，则说明该公钥已被使用，认为签名交易不合法；若没有出现过，则认为签名交易合

法，并将I加入集合$\mathcal{I}$。

3.2 一次性账户系统

3.2.1 一次性账户系统构成

一次性账户系统是整个隐私交易的基础，系统中每个用户拥有唯一主账户和多个子账户，子账户也可以看作智能合约中的账户，而子账户通常不是用户自己产生，而是由交易对方为用户生成。

Alice 拥有唯一主账户(A,B)，以及若干子账户(A_1,S_1)，(A_2,S_2)，(A_3,S_3)，…。Bob 为 Alice 转账时，通过主账户(A,B)生成了(A_1,S_1)和(A_2,S_2)，Carol 为 Alice 转账时，生成了(A_3,S_3)。可以看到为 Alice 转账的每一笔交易都会给 Alice 生成一个子账户，即 Onetime-account，而只有 Alice 拥有这些子账户的管理和使用权限。

3.2.2 账户生成算法

$E(F_p)[r]$是椭圆曲线E/F_p的r阶子群，G是$E(F_p)[r]$的生成元。E/F_p选取比特币或以太坊的曲线$y^2=x^3+7$。

- **主账户生成**

Alice 在 Wanchain 上原有账户为(A,a)，其中$A=[a]G$，为生成一次性账户系统主账户，则选取随机数$b\in[1,r-1]$，计算$B=[b]G$，以(a,b)作为 Alice 的私钥，以(A,B)作为 Alice 主账户公钥。最终 Alice 拥有主私钥(a,b)和扫描密钥(A,b)，而公开(A,B)作为 Alice 的主账户地址。

- **子账户生成**

当 Bob 要为 Alice 转账时，需要使用 Alice 的主账户(A,B)为 Alice 生成子账号(A_1,S_1)。

Bob 产生随机数$s\in[1,r-1]$，计算

$$S_1=[s]G$$

$$A_1=A+[\mathrm{Hash\_p}([s]B)]G$$

这里Hash_p 是将椭圆曲线的点映射到$[1,r-1]$的函数。

则 Alice 生成子账户为

$$(A_1, S_1)$$

(A_1, S_1)是一次性子账户，由于随机数s并没有公开，只是公开了含有s信息的S_1，而S_1并不能计算出s，由椭圆曲线离散对数问题保证。A_1是子账户的公钥分量，S_1是随机因素分量。

Alice 的每一个子账户都是基于主账户(A, B)的随机账户。

- **子账户的验证和对应私钥**

Alice 扫描链中所有一次性账户，取链中的S_1分量和扫描密钥(A, b)参与计算

$$A_1' = A + [\mathrm{Hash}([b]S_1)]G$$

若$A_1' = A_1$，则可以确定(A_1, S_1)是自己的子账户；

若$A_1' \neq A_1$，则可以确定(A_1, S_1)不是自己的子账户。

这是由于$B = [b]G$，于是

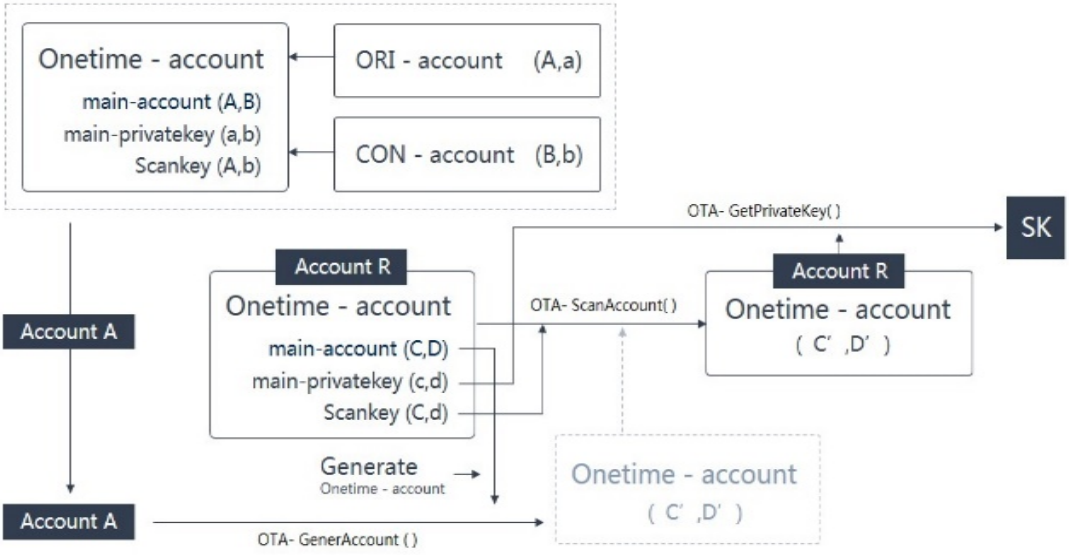
$$[s]B = [sb]G = [b][s]G = [b]S_1$$

由于仅有 Alice 知道扫描密钥(A, b)，则只有 Alice 能验证属于自己的子账号(A_1, S_1)。

Alice 需要花子账号(A_1, S_1)的资产，需要计算(A_1, S_1)对应的私钥x，

$$x = a + \mathrm{Hash}([b]S_1)$$

于是有子账户的公钥分量$A_1 = [x]G$。对于每个子账户，Alice 都可以计算其对应的私钥x。



3.3 基于一次性账户的邮票系统

前文中发现，当使用环签名进行隐私保护时，交易的发起方不可追溯，那么将导致交易费不知从哪个账户收取的问题 ,为解决这个问题 ,在 Wanchain 上设计并实现邮票系统。邮票系统基于一次性账户体系，在邮票系统中，每张邮票就是一个一次性账户，邮票系统中使用$Onetime-stamp$表示，实际上$Onetime-stamp$与$Onetime-account$相同。用户需要使用代币隐私交易的时候就需要预先购买邮票，将邮票“贴”在交易上才可以完成隐私交易，每张邮票只可使用一次，下面进行详细说明。

邮票系统是一个智能合约，其中设置若干面值，每种面值下存储用户购买的相应面值的邮票信息。合约中提供购买邮票和退还邮票功能：

购买邮票功能：购买邮票功能为用户提供使用万币购买邮票的服务。若用户$Account$需要购买邮票，则发起一笔交易，向邮票系统智能合约转账预购邮票面值$value$的万币并调用合约中购买函数，参数是用户为自己生成的一次性账户，而这一账户将作为邮票$Onetime-stamp$存储在智能合约面值$value$的列表之中，表示用户已经成功购买了这张邮票：

$$Tx = (Account, StampSC, value, payload, sig)$$

$$payload = ("purcharse", Onetime-stamp)$$

退还邮票功能 :退还邮票功能为用户提供将未使用的邮票退款的服务。若用户$Account$已购买邮票$Onetime-stamp$，且邮票未使用，则用户可调用邮票系统智能合约中的邮票退还函数将相应面值万币退还到自己的账户$Account$。退还函数所需参数为邮票$Onetime-stamp$、邮票面值$value$和邮票与自己所做环签名$ringsig$，退还交易确认后合约将在相应面值列表中删除这张邮票，表示邮票已退还：

$$Tx = (Account, StampSC, payload, sig)$$

$$payload = ("refund", Onetime-stamp, value, ringsig)$$

$$ringsig = (I, c_1, c_2, r_1, r_2)$$

在邮票退还功能中之所以使用环签名是为了要求用户提供邮票所对应的I值，以保证邮票并未使用过，不会出现邮票使用后再退还的情况。

实现邮票系统后，若用户需要发起隐私保护的代币交易，则先行在邮票系统中购买邮票，邮票面值基于调用合约消耗和用户意愿而定，然后在代币交易的Tx中不在出现用户的账户信息，而将用户使用的邮票与随机选取同面值的邮票形成集合作为交易发起方，以邮票环签名作为交易合法性的保证：

$$OTA\_Tx = (StampSet, TokenSC, payload, ringsig_{StampSet})$$

代币交易扣除的交易费就是所使用邮票的面值，所以每张邮票只能使用一次，而矿工在挖出新区块后，其挖矿奖励除了$Coinbase$原有部分外，还需扫描区块，将隐私保护的代币交易中使用的邮票价值一并给予，这样一来$Coinbase$比原有情况增加了邮票价值的量。为了保证系统万币总量不变，我们设置邮票智能合约的地址为固定的数值，如$WANChainStampSystem$的 hash 结果，以此保证任何人没有合约的私钥，也就无法将合约收到的万币转出（除用户主动申请退还邮票外），这就意味着邮票系统合约中的钱被锁死，在$Coinbase$增加的情况下保证了万币总量不变。

3.4 原生币交易隐私保护方案

原生币的交易隐私保护方案类似于邮票系统的实现方式，通过在 Wanchain 上部署一个对应于万币智能合约实现。

3.4.1 万币智能合约

万币智能合约是 Wanchain 上部署的一个类似于邮票系统的智能合约，其中设置若干离散面值的代币，代币以 1：1 比例对应于万币，智能合约中提供两个功能：购买代币功能和退还代币功能。购买代币功能允许用户向合约转等值万币，合约中将把用户提供的一次性账户添加到相应面值的存储列表之中；退还代币功能允许用户以一次性账户的环签名为参数进行调用，成功后将相应面值万币退还到用户的账户之中，下面根据交易场景详细说明万币交易的隐私保护过程。

3.4.2 交易场景

用户 1 主账户私钥为(a,b)，主账户公钥为(A,B)；用户 2 主账户私钥为(c,d)，主账户公钥为(C,D)。用户 1 希望为用户 2 转移$value$的万币。此为交易的场景。

3.4.3 交易流程

- **交易发起**：

用户 1 首先利用用户 2 的主账户公钥(C,D)为其构造一次性账户$Onetime-account2$，然后以一次性账户为参数调用万币只能合约$WancoinSC$的购买代币函数：

$$Tx=(Account1,WancoinSC,value,payload,sig)$$

$$payload=("purchase",Onetime-account2)$$

- **交易发起确认**：

Validator 接收到这笔交易，验证sig与$Account1$对应关系，验签通过后调用万币智能合约，合约把$Onetime-account2$存储在对应于$value$面值下的列表中。

- **交易接收**：

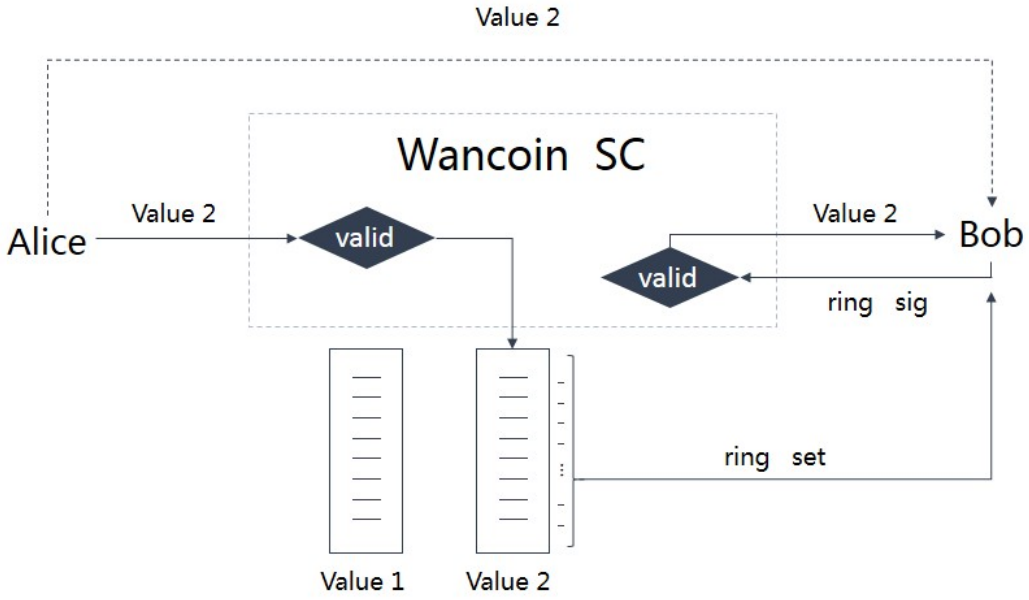
用户 2 利用其扫描密钥扫描$WancoinSC$的存储列表，发现$Onetime-account2$属于自己，然后利用$Onetime-account2$在其对应面值列表中随机选取$n-1$个一次性账户做环签名，并把环签名作为参数调用合约中退还代币函数：

$$Tx=(Account2,WancoinSC,payload,sig)$$

$$payload=("refund",OTASet,ringsig)$$

- **交易接收确认**：

Validator 接收到这笔交易，验证sig与$Account2$对应关系，验签通过后调用万币智能合约，合约验证环签名合法性，并通过$ringsig$中I值未出现过确认该账户没有提过款，确认无误后将向$Account2$账户转入$value$面值的万币。



3.4.4 隐私效果分析

基于 Onetime-account 和环签名的万币隐私交易方案能够做到以下隐私效果：

1) 利用一次性账户，交易发起过程中，无法得知交易发起者将代币转给哪个用户。
2) 利用环签名，交易接收过程中，无法得知接收者提出的是哪个一次性账户中的代币，但同时保证每个一次性账户不会重复提款。

在本方案中，交易发起方可以与购买代币的一次性账户建立联系，交易接收方无法与一次性账户建立联系，就保证了发起方与接收方之间不能对应，也就实现了隐私保护，但目前为了便于环签名的使用，万币智能合约中只能设置几个固定离散面值的存储，无法实现任意面值的交易，这也将是后续研究中的一项。

3.5 代币交易隐私保护方案

3.5.1 交易场景

用户 1 主账户私钥为(a, b)，主账户公钥为(A, B)；用户 2 主账户私钥为(c, d)，主账户公钥为(C, D)。用户 1 在智能合约 SC 中账户为$Onetime - account1$，想要从这个账户

中为用户 2 转移$value$的代币。此为交易的场景。

3.5.2 交易流程

交易发起

用户 1 要发起这笔交易，需先购买邮票，记为$Onetime-stamp$，然后构造一笔合法的交易 ,通过 P2P 网络将这笔交易传播出去。一笔交易包含四个字段 :TransFrom、TransTo、Data、RingSig。构造过程如下：

交易发起流程

1) 在邮票系统中随机选取$n-1$个与$Onetime-stamp$同面值的邮票 ,合同$Onetime-stamp$构成一个含有n张邮票的集合$StampSet$，构成整笔交易字段 TransFrom。这是为环签名做准备，隐藏交易发起者身份。节点P_i通过安全信道将$f_i(j)$发送给P_j，$j=1,\dots\dots,n$

2) 交易字段 TransTo 为调用的代币智能合约的地址$TokenSC$

3) 交易字段 Data 同样包含四个字段：SC_TransFrom、SC_TransTo、SC_Value、SC_Sig。构造过程为：首先为用户 2 构造$Onetime-account2$；交易输入 SC_TransFrom 为用户 1 控制的$Onetime-account1$，交易目标地址 SC_TransTo 为刚为用户 2 构造的$Onetime-account2$，交易代币值 SC_Value 为$value$，并且用户 1 为这笔交易计算签名sig，它与$Onetime-account1$匹配。

4) 交易字段 RingSig 的构造 :使用环签名方案 ,将 Trans From 作为公钥集合对交易的 Trans From、Trans To、Data 进行环签名，得到签名$ringsig$。

5) 最终代币隐私交易的交易结构为：

$$OTA\_Tx=(StampSet,TokenSC,Data,ringsig)$$

$$Data=(Onetime-account1,Onetime-account2,value,sig)$$

交易验证

Validator 在拿到一笔交易后，进行如下验证以及计算：

交易验证流程

1) 验证交易中 RingSig 是否与 Trans From 的账户集合匹配，如果验证通过，则表明交易合法，否则拒绝

2) 以 Data 字段作为输入，执行 TransTo 中地址对应的智能合约。智能合约会验证 SC_Sig 是否与 SC_TransFrom 匹配，如果匹配则在 SC_TransFrom 账户中减掉 SC_Value 的代币，同时创建 SC_TransTo 账户，并添加金额 SC_Value。

交易确认

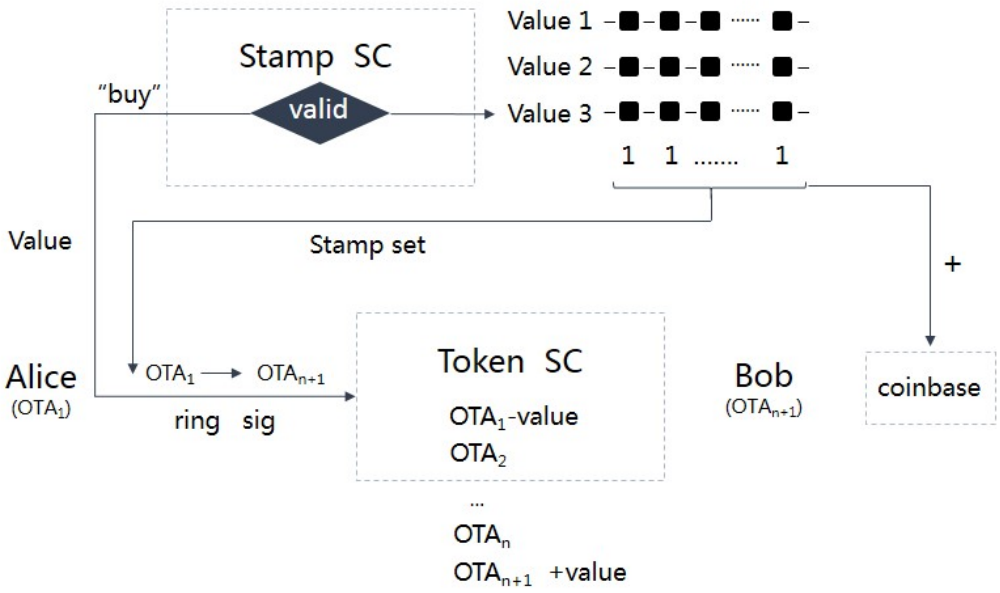
用户 2 在需要确认用户 1 的代币转移是否成功：

交易确认流程

1) 利用扫描密钥对智能合约维护的 Onetime-account 进行扫描

2) 扫描发现$Onetime-account2$属于自己，然后利用主私钥和$Onetime-account2$计算其对应的私钥

3) 至此，用户 2 已经确认收到一笔代币转账，并获得新的一次性账户$Onetime-account2$的使用权限。



3.5.3 隐私效果分析

基于 Onetime-account 和环签名的隐私交易方案能够做到以下隐私效果：

1) 利用邮票系统和环签名方案保证交易发起者全网匿名。
2) 利用 Onetime-address 保证智能合约代币账户与主账户隔离。

在本方案中，为将交易发起方隐匿，使用邮票系统和环签名，邮票系统中邮票与用户的对应关系是可追溯的，但隐私交易时对邮票进行了环签名，使得每笔隐私交易所使用的邮票是不可追溯的，也就将交易和真实发起方隔离，任何人无法追溯交易真实发起者，进而实现对发起方的隐私性保护。对于交易接收方，在智能合约中使用一次性账户系统，使得每次转账交易都为接收方创建新的一次性账户，任何人在没有接收方扫描密钥的前提下无法确认一次性账户的归属关系，进而实现了对接收方的隐私性保护。

-- 参与黄皮书编写 --

吕旭军（Jack Lu）、肇中、石榴、Zane Liang 、张英、杨涛、
Eric Swartz、陆利华

白皮书审阅：沈波、Dustin Byington、David A. Johnston、Michael Y.、
Jian Shen 、韩峰、 Albert Ching

官方网址/ wanchain.org　获取项目详情/ info@wanchain.org