计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210200129

一种基于区块链的融合 DKG 与 BLS 的高阈值签名协议

刘　峰[1,2,3]　王一帆[2,4]　杨　杰[2,3]　周爱民[1]　齐佳音[2]
1 华东师范大学计算机科学与技术学院　上海 200062
2 上海对外经贸大学人工智能与变革管理研究院　上海 200336
3 华东师范大学交叉创新实验室　上海 200062
4 上海墨珩网络科技有限公司　上海 200060
(lsttoy@163.com)

摘　要　阈值签名协议是多方信息安全协议的基础工具之一,在身份认证、防伪防抵赖等方面有着广泛的用途。文中研究了一种新型的去中心化阈值签名协议(Blockchain-based High-threshold Signature Protocol,BHTSP),通过将分布式密钥生成协议(Distributed Key Generation,DKG)与 BLS 签名(Boneh-Lynn-Shacham Signature)相结合,设计了一套可多方参与的、签名长度固定的阈值签名协议。在协议的实现过程中,采用了区块链智能合约作为协议的通信层,以确保协议参数的安全交换。仿真实验结果表明,BHTSP 协议可以产生固定体积的阈值签名,且存储验签的公钥组合需要的平均内存消耗相比 Schnorr 签名方案减少了 85.3%。在实验的区块链平台中,BHTSP 能够支持多达 50 个参与方参与的阈值签名生成,优化了区块链多方参与交易的执行流程。

关键词:区块链;智能合约;DKG 分布式密钥生成;BLS 签名;阈值签名;多方安全计算
中图法分类号　TP309.2

Blockchain-based High-threshold Signature Protocol Integrating DKG and BLS

LIU Feng[1,2,3],WANG Yi-fan[2,4],YANG Jie[2,3],ZHOU Ai-min[1] and QI Jia-yin[2]
1 School of Computer Science and Technology,East China Normal University,Shanghai 200062,China
2 Institute of Artificial Intelligence and Change Management,Shanghai University of International Business and Economics,Shanghai 200336,China
3 Cross Innovation Laboratory,East China Normal University,Shanghai 200062,China
4 Moheng Tech. Inc.,Shanghai 200060,China

Abstract　Threshold signatures are fundamental tools for multi-party information security protocols. It is widely used in fields such as identity authentication, anti-counterfeiting and tamper-resistance. We introduce a new decentralized threshold signature protocol BHTSP which combines distributed key generation (DKG) and BLS signature. The protocol allows multi-party participation and generates a signature of constant size. We implement this protocol with smart contract as the communication layer for secure parameter exchange. Experimental simulation results show that BHTSP can generate threshold signature with constant size. It reduced the memory consumption for aggregated public key combinations needed in signature verification by 85.3% compared to Schnorr signature. In the experimental blockchain platform, BHTSP is able to support the generation of threshold signatures involving up to 50 participants, optimizing the execution process for blockchain multi-party transactions.

Keywords　Blockchain, Smart contract, DKG-distributed key generation, BLS signature, Threshold signature, Secure multi-party computation

1　引言

自 2008 年比特币[1]问世以来,经过 10 余年的发展,区块链系统作为一种分布式的、链上数据难以被篡改的数据存储系统,已被认为是未来数字领域最具有潜力、最具颠覆性的技术之一。2014 年以太坊[2]的出现更是将区块链概念从传统金融支付领域扩展到了通用计算领域,其内置智能合约编程语言可在以太坊虚拟机环境中支持绝大多数的计算[3]。大规模分布式应用的发展,也是从此时开始兴起的。在广泛的应用场景中,多方参与的区块链应用系统交易流程的改进与优化,尤其是在一些支持多方参与的数字签名上的研究与发展,近年来备受瞩目。

到稿日期:2021-02-21　返修日期:2021-05-27　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(72042004)
This work was supported by the National Natural Science Foundation of China(72042004).
通信作者:齐佳音(qijiayin@139.com)

实现传统的多方参与的阈值（门限）签名协议，通常需要从基础的分布式系统通信开始，逐层向上地实现协议流程[4]。其中，分布式密钥生成协议（Distributed Key Generation，DKG）与分布式系统共识的实现复杂度较高，且不具备普适性。因此，虽然阈值签名相关的研究已经有不少的成果，如基于 Schnorr 签名的 Musig 协议[5]、基于 ECDSA 签名假设的阈值 ECDSA 方案[6]等，但可靠且能复用的工程实现却依旧很少。本文在现有阈值签名的基础上做出了如下贡献：

（1）将 DKG 协议与 BLS 签名[7]相结合，设计了一套去中心化的阈值签名协议 BHTSP，解决了传统的基于椭圆曲线的阈值签名协议中代数结构复杂、无法聚合签名、协议实现难度大等问题。

（2）使用区块链智能合约作为上述协议的通信层，以实现安全的协议参数交换。利用底层区块链提供的拜占庭容错机制，降低上层 DKG 协议实现的难度，提高了协议的安全性，从签名协议上优化了区块链系统的交易流程。

（3）在参与方人数较多（人数大于等于 6）的高阈值（阈值大于等于 4）签名场景中，设计的 BHTSP 协议在内存空间以及 Gas 消耗上显著优于常见的 ECDSA 多重签名方案与 Schnorr 聚合签名方案，且最多能支持 50 个参与方进行签名的场景。

本文第 2 节介绍了智能合约与 Gas、DKG 协议、常见阈值签名等基础知识；第 3 节阐述了 BHTSP 阈值签名协议的流程与关键步骤；第 4 节基于设计的 BHTSP 阈值签名协议进行了仿真实验，展示了协议仿真代码的运行结果以及相应的智能合约实现，并对结果进行了分析；最后总结全文并展望未来。

2　相关知识

2.1　智能合约与 Gas

狭义的智能合约可看作是运行在分布式账本上的预置规则，具有可封装、可验证的特点，同时也是一种能够完成信息交换、价值转移和资产管理的计算机程序[8]。广义的智能合约则是无需中介、能够进行自我验证并能自动执行合约条款的计算机交易协议。区块链上的节点会储存和执行智能合约的代码，执行的结果会被记录在区块链上。由于智能合约的程序语言与其运行环境都具有确定性，因此保证了各个节点在执行智能合约程序时，在相同的输入参数下具有相同的输出结果，不会造成各个节点之间链上数据的不匹配。

为了能够灵活地解决各类计算问题，智能合约语言通常是图灵完备的[9]。但是，图灵完备的特性也造成了智能合约程序存在无法退出计算、代码陷入死循环的问题。因此，在设计运行智能合约的虚拟机时，通常都会引入类似超时等的约束机制，以便在上述情况发生时，各个节点上的虚拟机能够及时退出计算。Gas 则是以太坊在超时概念的基础上，结合区块链的金融属性而设计的一种保护区块链节点计算资源的约束机制。

区块链上的每一次智能合约的调用，都需要显式地指定 Gas 消耗数量的上限和单位 Gas 的价格上限。两者的乘积决定了这次智能合约调用的成本上限。对于智能合约被编译后的字节码，每一步被执行时，都会消耗一定数量的 Gas。当智能合约实际执行中累计消耗的 Gas 数量超过调用时指定的 Gas 数量上限时，虚拟机就会退出智能合约的计算。因此，在智能合约的执行环境中，每一次合约调用所能承担的计算量上限是一定的，并且需要用户自身为此负担一定的经济成本。每一次智能合约调用需要使用的 Gas 数量与区块的总 Gas 的限制，也决定了该智能合约方案的扩展性和经济性。

2.2　DKG 分布式密钥生成协议

DKG 协议，即分布式密钥生成协议，指在参与协议的多个节点之间共同协作产生一组密钥的分布式协议。VSS（Verifiable Secret Sharing）协议，即可验证秘密分享协议，是 DKG 协议的重要理论基础。VSS 指在多方之间进行一个秘密数据的分享时，可以在不泄露秘密数据本身的情况下，将秘密数据拆分成多个片段，交由多方共同保管。之后，规定一个阈值，当需要还原秘密数据时，需要收集超过阈值的片段数量才能还原成功。如在 3 方参与的交易中，约定阈值为 2，只有大于等于 2 方的人意见一致才能还原秘密，否则视为失败。VSS 协议最早是由 Shamir 于 1979 年提出的，是一个基于多项式的秘密分享协议[10]。该协议首先构造了一个多项式：

$$f(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n \tag{1}$$

其中，a_0为想要分享的秘密数据。然后任取 $n+1$ 个该多项式上的点，并将这些点在参与协议的节点之间进行分享，任何少于 $n+1$ 个点都无法推断出原始的秘密数据，只有当获得全部 $n+1$ 个点之后，才能通过重构多项式系数的方式还原秘密数据a_0的值。

在此基础上，第一个 DKG 协议是由 Pedersen 于 1991 年提出的[11]。之后该协议被多次修改并应用于门限密码学及其应用的研究，近年来 Daouda 等就借助了计算 Diffie-Hellman 假设设计了能应用于多方的密钥生成与交换协议，以此来拓宽门限 ECDSA 签名使用范围[12]。但是将 DKG 协议应用至区块链上时存在一个问题，当 DKG 的各个参与节点得到超过阈值的秘密片段时，所有参与节点都可以独立地恢复出秘密的明文。由于这一秘密在一轮的秘密片段交换后即暴露，因此无法在下一轮的加密活动中被使用，即该算法所能够提供的秘密分享是一次性的。如果需要 n 次的秘密分享，则需要将协议执行 n 次，效率较低。

1999 年，一个改进的协议由 Rabin 等提出，被称为 Joint-Feldman 协议[13]。该协议的基本思想是并行执行 n 次 Feldman 的可验证秘密分享协议，其中每一位参与者都在本地产生一个随机的多项式，然后把随机选择的秘密值在所有参与者中分享。由于分享的是秘密的一个承诺（commitment）而不是秘密本身，因此只要不发生超过阈值 t 的多人共谋作弊，就不能恢复该公钥对应的全局私钥。

从 Joint-Feldman 协议的流程可以看出，利用该协议在阈值与秘密承诺等方面的性质，结合与之相匹配的签名算法，就可以用于构造分布式的阈值签名协议。

2.3　ECDSA 签名与 Schnorr 签名

传统的基于椭圆曲线的 ECDSA 签名算法被广泛应用于签名领域，该签名算法主要是从非奇异的椭圆曲线上选择一

个基点，用于产生公私钥，再借助随机数对哈希处理过后的消息进行签名。整个算法的安全基于离散对数难解问题，但该签名算法无法直接实现签名聚合。因此，该算法的签名聚合方案需要间接地逐个产生签名，然后对签名进行逐个验证。最终签名长度会随着阈值的增加而增加，在多方参与的场景中劣势明显。

与此相对应，在已有的多方阈值签名中，Schnorr 聚合签名是一种计算量小且签名快的阈值签名方案。以三方为例的聚合签名 Mul_{sig} 的计算式如下：

$$Mul_{sig}=\sum_{i=1}^{3}r_i+H(m|\sum_{i=1}^{3}R_i|\sum_{i=1}^{3}Pb_i)*\sum_{i=1}^{3}Pr_i \tag{2}$$

其中，(r_i,R_i) 为单个参与方用于签名的随机数对，m 为需要签名的消息，(Pb_i,Pr_i) 为单个参与方用于签名的公私钥对。从式(2)中不难看出，Schnorr 聚合签名具备加同态特性，且签名方式简便。但在这种线性签名公式中，签名时需要验证的公钥数量也会随着参与人数的增多而增多。此外，Schnorr 签名在聚合时需要多轮通信，在实际部署时会比较困难[14]。

2.4 BLS 签名

基于双线性映射的 BLS 签名算法的聚合签名方案可以产生短签名，即聚合后的签名长度与单个签名长度相同。已有多个与区块链相关的改进方案采用了基于双线性映射的聚合签名技术[15]。

BLS 签名协议是由 Boneh 等于 2001 年提出的[16]。该签名的核心是双线性映射(bilinear maps)函数。BLS 签名在离散对数问题的基础上，在 3 个 q 阶循环群 G_1,G_2,G_T 中定义了一个双线性映射函数 e：

$$e:G_1*G_2\rightarrow G_T \tag{3}$$

如果函数 $e(P,Q)$ 同时满足式(4)中的性质，则 $e(P,Q)$ 就是一个双线性映射函数，其中 $a,b\in F_q^*$。

$$\begin{cases}e(P,Q+R)=e(P,Q)*e(P,R)\\e(P+S,Q)=e(P,Q)*e(S,Q)\\e(a*P,b*Q)=e(P,Q)^{a*b}\end{cases} \tag{4}$$

即配对函数对曲线上的运算满足交换律、结合律以及式(4)中的第 3 条等式。

在 BLS 签名算法中，私钥为一个秘密的整数 sk，公钥为 $pk=sk*G$，对于消息 m，将其映射为曲线上的一个点，记为 H，则签名 $S=sk*H$。在签名的验证过程中验证 $e(P,H)=e(G,S)$ 是否成立。借助椭圆曲线双线性配对函数特性可推导出其验签过程为：

$$e(P,H)=e(sk*G,H)=e(G,sk*H)=e(G,S) \tag{5}$$

将该算法用于签名聚合时，聚合后的签名 $S_{all}=\sum_{i=1}^{n}S_i$，其中每个子签名 S_i 都是不同私钥对同一个消息的签名。通过带入签名函数以及式(5)，可以推导出聚合签名的验签过程为：

$$\begin{aligned}e(S_{all},G)&=e(\sum_{i=1}^{n}S_i,G)\\&=e(sk_{all}*H,G)\\&=e(\sum_{i=1}^{n}sk_i*H,G)\\&=e(H,\sum_{i=1}^{n}pk_i)\\&=e(H,pk_{all})\end{aligned} \tag{6}$$

通过式(6)可以看出，BLS 签名算法对公钥、私钥与签名等密码学元素具有线性组合的能力。得益于这样的代数性质，其可以结合 DKG 中的 Joint-Feldman 协议，在不暴露全局私钥 sk_{all} 的情况下，使用全局公钥 pk_{all} 对消息 m 的聚合签名 S_{all} 进行验证，从而实现 BHTSP 阈值签名协议流程。

3 基于区块链的融合 DKG 与 BLS 的高阈值签名协议 BHTSP

3.1 BHTSP 协议流程

下文具体描述产生阈值签名的协议流程。假设有 n 个参与方，阈值为 t，$t<n$。

(1)参与方 V_i 需要在本地产生一个随机的多项式 f_i，多项式的阶为阈值 t，系数 $a_{i,j}$ 均为随机数，其中 $i=1,\cdots,n$：

$$f_i(x)=a_{i,0}+a_{i,1}x+a_{i,2}x^2+\cdots+a_{i,t-1}x^{t-1} \tag{7}$$

(2)每个参与方 V_i 广播 $A_{i,k}=g_2*a_{i,k} \bmod p$，$k=0,\cdots,t-1$，其中 p 为 BLS 循环群 G_2 的阶，g_2 为生成元，并指定 $r_i=a_{i,0}$，$R_i=A_{i,0}$。

(3)每个参与方 V_i 计算片段 $s_{i,j}=f_i(j) \bmod p$，$j=1,\cdots,n$，并将 $s_{i,j}$ 加密后发送给 V_j。

(4)每个参与方 V_j 验证步骤(2)、步骤(3)得到的 $A_{i,k}$ 与 $s_{i,j}$，需满足以下等式：

$$g_2*s_{i,j}=\sum_{k=0}^{t}A_{i,k}*j^k \bmod p \tag{8}$$

(5)如果验证不通过，则 V_j 可以举报 V_i 发送了错误参数。

(6)假设验证均通过，则参与方 V_i 还原全局公钥与 V_i 的本地私钥等协议参数。全局公钥为：

$$pk_{all}=\sum_{i=1}^{n}R_i \bmod p \tag{9}$$

节点 V_j 的本地私钥 sk_j 为：

$$sk_j=\sum_{i=1}^{n}s_{i,j} \bmod p \tag{10}$$

全局私钥 sk_{all} 在理论上可以通过计算得出，但协议中节点并不分享全局私钥的片段 r_i，因此在不超过阈值 t 个节点共谋作弊的情况下，任何节点都无法还原全局私钥。故仅给出全局私钥的还原公式：

$$sk_{all}=\sum_{i=1}^{n}r_i \bmod p \tag{11}$$

(7)当节点对消息 m 签名时，计算消息 m 的 $hash$：$h=h(m)$，节点 i 的本地签名为 $sig_i=sk_i*h$，将该签名广播至各个节点，则最终的阈值签名 Sig_{all} 可由拉格朗日插值计算得出。

$$Sig(x)=\sum_{i=1}^{n}sig_iL_i(x) \tag{12}$$

此处还原的 $Sig(x)$ 为签名对应的多项式，$L_i(x)$ 为插值基函数，阈值签名 Sig_{all} 即为该多项式的首项系数。

(8)验证上述签名时，需满足以下 BLS 验签公式：

$$e(Sig_{all},G_2)=e(h,pk_{all}) \tag{13}$$

以上就是结合了 DKG 协议与 BLS 签名后的阈值签名协议流程。图 1 给出了 3 个节点参与的协议流程。

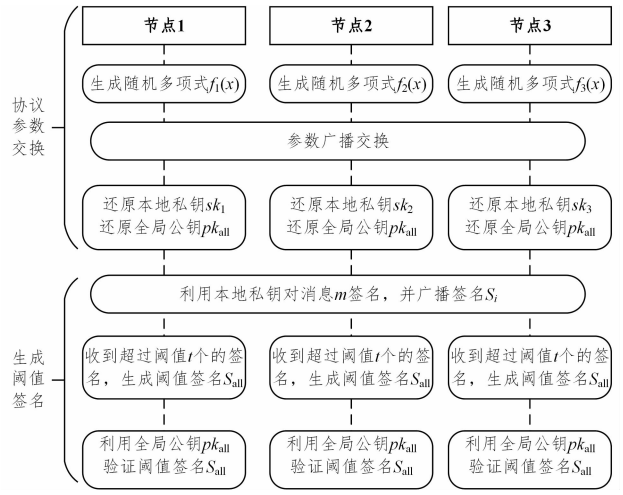

图 1　BHTSP 协议的流程示意图

Fig. 1　Flow diagram of BHTSP protocol

3.2　BHTSP 协议与智能合约的结合

对于上文中阈值签名协议的实现，其中一个难点是参与协议的多个节点在复杂的网络环境中的 DKG 协议参数交换。在实际的生产环境中，节点可能处于下线、宕机以及被攻击等非正常运行的状态，协议流程可能因此而失败，故需要寻找合适的方案来保证协议参数交换的安全性与时效性。

在现有的多方阈值签名协议的实现中，多数协议采用 P2P 网络作为协议参数广播的手段，其协议流程需要实现节点间对数据的共识协议。在处理异步网络通信等复杂场景时，现有协议代码的复杂度较高，并且缺乏对全局状态的存储。而区块链技术已经实现了一套 P2P 网络中多个节点之间对链上数据的共识协议。如果以区块链技术为基础，通过智能合约的方式来实现协议流程中参数的合理交换与自动化[17]，则可以降低协议实现的复杂度，实现上层阈值签名协议与下层节点共识协议等模块的解耦。

此外，有学者认为区块链上承载的数据具有支持审计的特性，可以有效降低数据被伪造与篡改的可能性[18]。还有学者将区块链智能合约的使用方式与公告栏进行了对比[19]，认为在智能合约中对数据进行交互，是一种有效的、公开且透明的数据通讯信模式。从智能合约的上述特性可以看出，该技术可以满足本协议中节点间协议参数交换的安全性与正确性要求。

据此，本文设计了一套用于 BHTSP 协议中参数安全交换的智能合约。如图 2 所示，该合约主要实现的流程为：1）节点注册流程；2）协议参数交换流程；3）异常节点（恶意节点、慢节点等）处理流程。

3.2.1　节点注册流程

参与节点需要在协议流程开始前注册自己的身份。由于区块链上数据的公开透明性，参与节点需要提供自己的地址以及用于加密链上数据的公钥。其他参与节点向该参与节点发送参数时，需要使用该公钥进行加密。此时，如果在合约设计中加入了对异常节点的经济惩罚方案，则参与节点在注册时需要同时支付一笔押金到智能合约中。注册完成后，参与节点激活自身的状态，开始执行协议流程。

3.2.2　协议参数交换流程

各个参与节点将本地产生的协议参数上传至智能合约。上传后，各个节点之间可查询自身相关的协议参数。需要注意的是，上传的参数分为两类，一类是公共参数，可以明文上传，另一类为私有参数，需用对方注册时提供的公钥加密上传。同时，上传者需要签名上传数据的 hash，以表明该协议参数的所有权。当发现链上的错误参数时，可以将此用作源头追溯。根据 3.2 节中的协议设计，需要向智能合约中上传的用于交换的两组协议公共参数为：

（1）多项式系数的承诺，即 3.1 节步骤（2）中的$A_{i,k}$；

（2）多项式的值，即 3.1 节步骤（3）中的$s_{i,j}$。

其中，参数（2）需要加密发送。各个节点下载上述参数后，用本地的私钥解密，可重构节点的本地密钥。

3.2.3　异常节点处理流程

如果重构本地密钥成功，则节点投票至智能合约，表明自己同意当下的协议参数。当合约中同意当下协议参数的节点个数大于阈值时，该协议参数随即生效。

如果本地重构不成功，如节点发现另一个参与节点有发送错误参数等行为时，则该节点会将作恶节点的参数作为证据上传到智能合约中。其他的节点可以对上传的错误参数证据进行验证，如果错误参数的确存在，则将作恶节点剔除到协议流程之外，并做相应的经济处罚。将协议流程中不能按时上传参数的节点称为慢节点，此类节点也会影响协议的正常运行，因此对于超过规定时间而没有上传协议参数的节点，也需要做剔除处理。

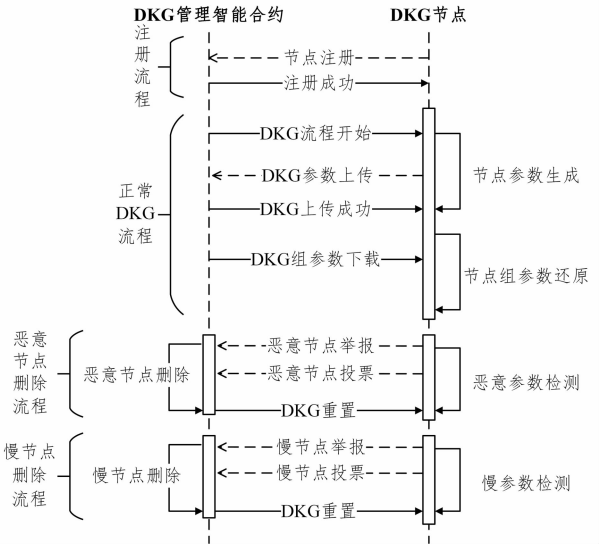

图 2　BHTSP 协议智能合约中的 DKG 管理流程

Fig. 2　Smart Contract-based DKG workflow of BHTSP protocol

4　仿真与实验

按照上述协议的设计，首先在本地环境中模拟了 BHTSP 协议的全部流程，从而验证了该协议的正确性与有效性。其次，利用相关区块链平台对智能合约代码进行了实际的测试，验证智能合约代码的扩展性和经济性。

实验测试运行的硬件环境如下：HP Z620 工作站；处理器为 8 核 Intel Xeon E5-2670@2.60 GHz；内存为 64 GB PC3-

12800R;硬盘为 WD 500GB SSD。

4.1 BHTSP 协议的正确性与有效性

本文运用仿真运行的方式来评估 BHTSP 协议的正确性和有效性。下文给出了仿真代码的运行结果,假设有 3 个节点参与该协议的运行,即 $n=3$,其中阈值 $t=2$,采用 BN256 曲线作为 BLS 签名中需要用到的双线性映射的具体实现。

(1)根据协议,每个节点都需要首先在本地随机产生一个 $t=2$ 阶的多项式,这里以多项式系数来表达,每个系数均为一个 256 bit 的整型,如下所示:

Polynomial #1:

a0=71c30cf428f4fad9469d546cfed35a2ea5605d825b2cabde88cf6ae4bdc1a5a5

a1=65973edd1fb34bd56a1fd1274e86734121106473f34063ea18fe3a702b87e298

a2=471d5bb272a67bd85468de6d6b83cd5629aa195dd2595c73e01e698a7ed20515

Polynomial #2:

a0=6d56d148c5bc592f92a533977c338356ad38e1fec38505f532343100bf6c41ae

a1=27c95bbf0c3c62adf97ea3d1465e53c82fc81846096c662378c785ada9401b9c

a2=84c23418cce3a362db92f58f66580a4de0ea9ab98332fbd13fa124382697411d

Polynomial #3:

a0=4f0f0a8781dea9eabff6a7fbd00c18d20e6cca371fe7df20533000fae931467d

a1=05b44dc663649c6c22e8ebe4953d7457c37d45d64ec211fe50ce4738d372b0a1

a2=6fa3faca27410648698486228c85b4aad4a06e488ae0d8f222d927f260C03f7b

(2)由上述随机多项式可以得到各个节点的私钥片段,即:

Private Share #1:

Private[1,1]:8ec2a5a070ab3a8d5ab617495758bea4c18d4d41289b331867bd1a84106f1af1

Private[1,2]:1a92f1eb086761fec2c0bd8fc3dc0584d3f35395aa6600f1d289b481b167b5a5

Private[1,3]:349df59a85708120d39d20b10766e71139ad8ca5d0e387b2fd93219450045a83

Private Share #2:

Private[2,1]:8a2d5f3d5438d746bd46e03fc765054b8f5e06eb57f92ec5d06de68b37972c06

Private[2,2]:01694fb99c9204369fbeb1ddbab80778a7afb7120a51a3cc9f5d0773f5eb4175

Private[2,3]:8229a8677eb277ec395c6e9a7abb1e4181d69cabc3101075ed8e70cd016dd91e

Private Share #3:

Private[3,1]:34b25134c1e0c4a5a1f42d4a904a65b377fcf043015f90ecaca87bcac5b7c438

Private[3,2]:69e88b9305c163f7ac8ad226080f3fc95c4064cd006dbb7931a452240c124e88

Private[3,3]:5efcb7bf02dcffe7354aa9d5d5d5caf28ca999c224e725a1c7f48fab6494730c

(3)还可以得到对应的公钥片段,即:

Public Share #1:

Public[1,1]:bn256.G2((145c8870ede859e4f9a03d00f6b66eddd4b89e684d851208a149db2f5b143038,60a52ca231d131f7d9ab8cbd847275c40848348810bce9690330e545ad61addd),(30bda319c465a0e12e1b1d2e92c87ecd8dbc21e1fe4c250d5eb8aab2f362fa9f,0b94f8907a736f0cfc922fb31d8fb36d1a0a7a44026304025b638ce8be96d0a6))

Public[1,2]:bn256.G2((3963cc0bd6e0f64f632890f777eea52d547732ed02f2262e2baa235d96929881,4bb0380754918fc9a046cdb333f490e66ab9ce04cb7f6efef44d79607122064a),(8191dcc6ce0fc465a4f6eaa8e785fc39ac7051802f7c4fe26949db3e934d0519,722e1eb311098a97dbb3e6f8e6c5e9963422077c4ff45179e6bdacdf7d7d1564))

Public[1,3]:bn256.G2((75e82f0ce48019047f5c6ed30fc6dd474c3ec0cd20bdf1f1079014e2e934d138,102de59471a0460014d52d9335b053b856410795bc7488eaace5a26aa2dfb739),(43b6cac30bf2cfbf02d9febd843f819b99c157b73f28e48bac20c135c94d14cf,4e46a497af0c2439c0b662ce636b4e61856216ba02c2c4324ac0f8d9f191572f))

Public Share #2:

Public[2,1]:bn256.G2((536c2667d3822b9ef66274217a502c2db5ad5c9dflbd7fc2f19912c88e5529e7,052a2946e683c70ff3c62d51848fc4cedc6805a6cde0c6baa4701863746ad1d6),(57143efabee61f3f0c2f1b6f024118b1481a2312025ed9df929afe0aefed7d8c,69f1e63a0f928459a0l1614e06738ac1854200b220b076107564000834d87c89))

Public[2,2]:bn256.G2((39b99aac8880fbd7ddb8f9725121dbcfec55e3e58e8e4380c5204ce51cf59741,2fa26cc4fb7d93600e0429941df822e35e1755cf2aa04ba66bde873aa122a65c),(32f15ea2368d6e494ce22d81b62ac35a7e7cd14da1c6cf0a526c62135771bb64,772926dbf7535ec3b7fd8607a83a8b13cae510777b4a3c10e896269aa373e6ff))

Public[2,3]:bn256.G2((173c6be6ea60b1521d412372ffe849cd9520e8d5ccb7e432343ac5f23195f8f0,46889b9834cdf22e047d41d08a70c7c474db2f51db51eb7c0183522bdc3ea8b8),(1f63fa348c35cf4e223263a82c6fc140526bb0cdd2963777b371e4d373510b53,2f5d084bc31c6565ce584d3fec6ef97e1a6fc7708a47ac9a5d5604eb26943ee0))

Public Share #3:

Public[3,1]:bn256.G2((7c34423764473ff6b7cc604794299b151bc2fe2a40faa2d55e2cb1d1ecfea1f6,859b94b5b17693211d37b652a5ad504f84929ec14027bb7815946dd1ae71e95c),(59f6fb73f33631fe3f9fd016a1f4e2a0aaa1d984884d25d2a64b55fac2c5a8b8,1f14e0a6c9e821eb95050f17baec20415fa12588deb1caebdeba7f990af1441d))

Public[3,2]:bn256.G2((243212ca8deb0ff486f15044d34d9887a843efbee4a10c789d2bf43ebfcc3e87,1539b50ed70284b8f

88681605d4054e69af534921dda9e5e9f31f2a25d663a95)，(0eae1ecddc497939b9ea0f11ebb2ec9bc79c1e949e4fca46bcebf1f152ad2a45，4c5b1ed9e61b3bd0ceca742642b613862961eca968bfa8085361498ed1c14c61))

Public[3，3]：bn256.G2((1cc3e0fec3aeb9752a3ceca1ed50709819b0a0e03bdb6aee4b3470f41a461e90，8bff11f4aae4887c6d2eb25e1af00e4a4a3b3babdd5329d87175cd00ab3914c4)，(6e8af13efbcaf219ef088b1a79605c8f635f5594e62ca2a5f473d91a35e2d0fb，753dcbdc0cdad51aa687b81518da9c4cb0c8fd857a720a92a41fe87e559a03ab))

(4)由此，可以重构出各个节点的本地私钥，即：

Local Private Key ＃1：

2e38524bf17dc68665114b62ebfe71616bcd2849919d8082b07594235e65266d

Local Private Key ＃2：

85e4cd37aabaca2d0f0a419386a34cc6d7e36f74b5256037a38b0e19b36545a2

Local Private Key ＃3：

860f53ddbc5c70fa97d44c68f672f42419a03500c0af84a698e72db15e5a344c

(5)假设对字符串“Hello World”签名，则各个节点的本地签名片段为：

Signature Share ＃1：

00001ffb8010502c44ebaf301d187bf62e8f60a5f385cf13d271adfd9aecd6b89b7511351cd47c7029311a109faf027a1f084c04a48d7208dbed1c67d37af8dfb0ce

Signature Share ＃2：

00011c0eac03f69df304a5b113853022849c5c7b4d71b422a71d551aeeb9ad43c8e26dc6160a4b3aedb2fa575ffad1b42dbf4eb4239d5c4ecaaab6624b3a66f400ed

Signature Share ＃3：

000256d6a4445fa9460131a6e18256893767ae8a48504aa6ad399f62c13eef223f206904f4645b50e969f5a6643ba8e2a44e6314726e081593aba7f6a9ba44a61f2e

(6)将本地签名片段聚合后，得到的全局签名为：

Group Signature 1：

409bc631d530392cfbff7f776191d4650bb4092ef71978ee17b7eb54d5f8ec482819b28c9b706693395f10d26de590ca4212d056a1a95e5f05d8798f81fddd4d

可以看到，聚合后的全局签名与单个签名的长度相同。

(7)通过(2)中的信息可以重构全局私钥：

Group Private key：

0ebee4fddb48ee004459568f88093e1503eaed924e431eabd9d5b429b706490e

(8)使用该全局私钥来签名(5)中的相同字符串，也可以得到全局签名：

Group Signature 2：

409bc631d530392cfbff7f776191d4650bb4092ef71978ee17b7eb54d5f8ec482819b28c9b706693395f10d26de590ca4212d056a1a95e5f05d8798f81fddd4d

可以看出，(6)中通过签名片段合成的全局签名与(8)中通过全局私钥直接签名消息而得到的全局签名是相同的，验证了协议的正确性与有效性。

4.2　BHTSP 协议与常见的阈值签名协议的比较

常见的阈值签名协议主要有两种，第一种协议基于 ECDSA 签名实现，并且已经被应用在比特币平台的多签方案(bitcoin multisig)上；第二种协议基于 Schnorr 签名实现，也已经被多个区块链平台作为签名聚合方案。在本次实验中，将上述两种阈值签名协议与 BHTSP 协议进行了对比，主要对比指标为：1)签名聚合后的长度；2)验签时对应的聚合公钥数量。

以 n/m 的阈值签名为例，根据拜占庭容错规则，实验中采用的阈值 $n=\lceil m*2/3 \rceil$。

从图 3 中可以看出，基于 ECDSA 的阈值签名方案的阈值签名长度会随着阈值的增加而增加，但基于 Schnorr 与 BLS 签名的阈值签名协议，其签名长度为固定的 64 byte。在高阈值 6/8 条件下，BHTSP 方案的签名长度仅为 ECDSA 方案的 16.6%。在实验的所有阈值组合中，BHTSP 方案的平均阈值签名长度相对于 Schnorr 方案降低了 73.1%。对于固定签名长度的 Schnorr 与 BLS 签名，两者在存储空间与网络带宽等方面比 ECDSA 方案更具备优势。

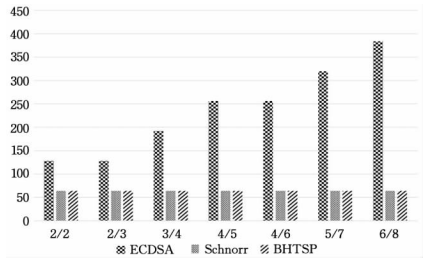

图 3　3 种阈值签名协议在不同阈值条件下的签名长度

Fig. 3　Signature size under various settings for protocols of ECDSA, Schnorr and BHTSP

在验签公钥的组合数量的比较中，由于 ECDSA 签名无法进行签名聚合，因此只比较基于 Schnorr 签名的阈值协议与本文的签名协议。实验中各协议采用相同的一组阈值。

从图 4 中可以看到，对于 Schnorr 聚合方案，其需要的用于验签的公钥组合数量较多，且随着阈值的增长而较快增长。如在 A，B，C 三方参与的 2/3 Schnorr 签名方案中，签名会出现 $(sigA+sigB)$，$(sigB+sigC)$，$(sigA+sigC)$ 的情况，对于这 3 种聚合签名的验证就需要准备 (pk_A+pk_B)，(pk_B+pk_c)，(pk_A+pk_c) 3 类公钥组合。以此类推，阈值越大，验签公钥数量就会越多。表 1 列出了在阈值大于 4/5 的情况下，Schnorr 方案需要的验签公钥组合数量。

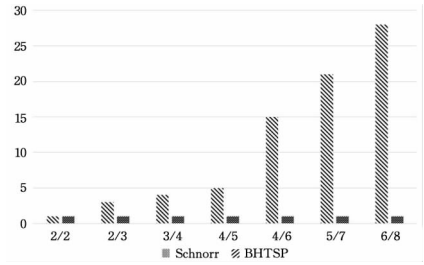

图 4　Schnorr 与 BHTSP 在不同阈值条件下需要的公钥组合数量

Fig. 4　Numbers of public key combinations under various threshold settings for Schnorr and BHTSP

表 1 高阈值下 Schnorr 聚合签名验签公钥组合测试数量

Table 1 Number of public keys combinations for Schnorr aggregated signature verification under high thresholds

阈值	验签公钥数量
4/6	15
5/7	21
6/8	28

与此相对应的是，BHTSP 协议可以使用同一个全局公钥来验签所有的阈值签名。在实验的所有阈值组合中，BHTSP 方案的平均验签公钥组合数量比 Schnorr 方案少 90.9%。

此外，实验中还统计了两者用于存储验签公钥组合的内存开销，如图 5 所示。尽管 BHTSP 的公钥单个大小为 128 byte，大于 Schnorr 方案的 80 byte，但由于 Schnorr 方案需要存储较多的公钥组合用于快速验签，因此在阈值大于 4/5 的情况下，Schnorr 方案的内存消耗较大，而 BHTSP 方案则可以保持固定的内存消耗。在阈值 6/8 条件下，BHTSP 方案的内存消耗仅为 Schnorr 方案的 5.7%。在实验的所有阈值组合中，BHTSP 方案的平均内存消耗比 Schnorr 方案低 85.3%。

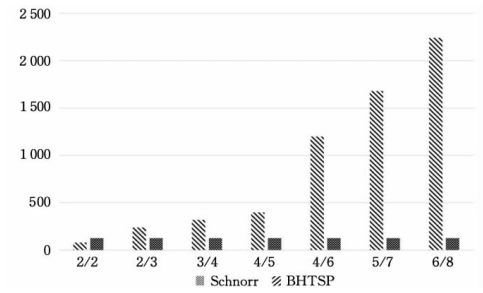

图 5 Schnorr 与 BHTSP 在不同阈值条件下的验签公钥存储内存消耗

Fig. 5 Related memory consumption under various threshold settings for Schnorr and BHTSP

综上所述，BHTSP 方案在阈值签名长度上优于 ECDSA 方案，与 Schnorr 方案相同；在验签的公钥组合数量上优于 Schnorr 方案。因此，在实际应用时，BHTSP 方案可以占用更少的内存空间与磁盘空间，在数据传输时减少带宽消耗。

4.3 基于区块链实现的扩展性与经济性

实验仿真代码设计中，参数交换在本地进行，在参数保护上无需考虑分布式系统中的恶意攻击。对于安全问题，仿真时使用相关区块链平台实现了 3.2 节中提到的智能合约，用于实验协议流程中参数的安全交换。

由于该区块链平台的区块总 Gas 数量被限制在 900 万，因此单个节点无法向智能合约发送大于 900 万 Gas 的参数进行交换调用。从图 6 可以看出，随着参与节点的增加，参数交换合约调用的 Gas 消耗逐步增长，最终被限制在 900 万。这时，能够参与协议的节点数量为 50 个。这也意味着，协议在区块链系统的多方交易场景中，具备极高的扩展能力。结合图 5 的验签公钥存储内存消耗低且固定的优势，BHTSP 也有很大的潜在经济价值。

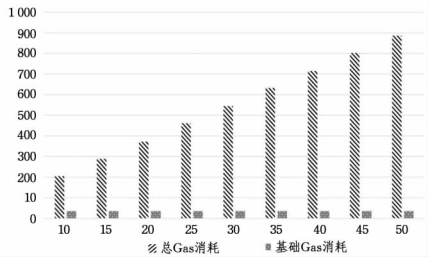

图 6 参与节点数量与参数交换的 Gas 消耗

Fig. 6 Gas fees for parameter exchange in regard to the number of participants

4.4 实验分析

4.4.1 BHTSP 协议的正确性与有效性分析

在 4.1 节的实验中，分别采用了两种方式来产生最终的阈值签名。在第一种方式中，仿真程序完整地执行了 3.1 节中介绍的 BHTSP 协议流程，最终通过拉格朗日插值的方式生成了阈值签名；在第二种方式中，利用了仅在实验环境中才能实现的全局私钥的重构，完成了对消息的签名。通过对生成的两组签名进行比较分析得出两者是相同的，从而验证了协议的正确性与有效性。

4.4.2 BHTSP 协议与常见阈值签名协议的比较

表 2 列出了 BHTSP 协议与常见的阈值签名协议的属性对比结果。从表 2 中可以发现，基于 ECDSA 的阈值签名方案由于代数结构较为复杂，无法对阈值签名进行聚合，因此其阈值签名的长度会随着阈值的增加而增加。相对应地，由于 Schnorr 方案与 BHTSP 两种签名算法本身都支持签名聚合，因此其阈值签名长度都可以控制在单个签名的长度上，即短签名。

表 2 3 种阈值签名协议的属性对比

Table 2 Comparison of properties by common threshold signature schemes in ECDSA, Schnorr and BHTSP

协议	是否支持签名聚合	聚合前是否需要通信
ECDSA 方案	否	不适用
Schnorr 方案	是	是
BHTSP 方案	是	否

此外，对于 Schnorr 方案，由于其签名算法中存在随机数 r，因此各方在产生本地签名前，需要首先交换一轮各自的 r 值，才能产生本地签名。进一步地，由这些本地签名聚合的阈值签名需要该 n 方的本地公钥聚合后的公钥才能验签，因此，在不同的 n 方组合下，对应的聚合公钥的总数量为 $m!/(m-n)!$，即 m 个参与方中有任意的 n 个参与签名聚合。

与 Schnorr 方案不同的是，BHTSP 方案中的阈值签名协议充分利用了上文提到的 BLS 签名算法可以对公钥、私钥、签名等进行线性组合的特性，在将其与基于多项式的 Shamir 秘密分享结合后，可以使用同一个聚合公钥来验证所有的阈值签名。

4.4.3 签名时间与智能合约 Gas 消耗分析

由于研究设计的阈值签名在聚合签名时，利用了拉格朗日插值算法来还原多项式的参数，根据式(12)与朴素拉格朗

日插值算法的实现，其时间复杂度为 $O(n^2)$[20]。可见该协议在大规模多方场景的时间开销上仍有待改进的空间。

对于智能合约的 Gas 消耗，根据 BHTSP 协议，单个节点需要上传到智能合约的阈值签名协议数据的增长是线性的，即 $O(n)$。同时也明晰了绝大部分的 Gas 消耗来自于协议数据在链上的存储。因此，可以推断出实验中单个节点的 Gas 消耗的增长也是线性的，即 $O(n)$。在图 6 中观察到的 Gas 消耗符合上述分析。

结束语　阈值签名技术是多方信息安全协议的基础工具之一，但传统方案的实现，特别是多方参与的方案的实现，具有一定的难度。区块链技术是密码学与分布式系统的一次完美结合。在传统的密码学和分布式系统领域，从零开始实现一套新的协议是比较困难的。但是智能合约的出现，特别是支持图灵完备语言的虚拟机运行环境的出现，显著降低了实现分布式协议的门槛。BHTSP 协议充分利用了区块链的优势，快速且安全地实现了一个可商用的分布式阈值签名协议，从签名长度、验签公钥内存消耗以及 Gas 消耗上进行了优化，减少了区块链多方交易的开销。本文提到的各类技术的具体实现，已经公开发布在墨客区块链平台上，以供大家参考。对于 DKG 协议的实现，除了 Joint-Feldman 协议外，还有另外几种实现方式，未来也会探索将其他的几种 DKG 协议与 BLS 签名相组合的方案，以丰富密码学和区块链技术的生态。同时，也会选择不同的区块链实验平台，突破上文中的 Gas 限制，获得更高的签名阈值。

参 考 文 献

[1] NAKAMOTO S. Bitcoin: A Peer-to-Peer Electronic Cash System[EB/OL]. https://bitcoin.org/bitcoin.pdf.

[2] BUTERIN V. A next-generation smart contract and decentralized application platform[EB/OL]. https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf.

[3] CHEN W L, ZHENG Z B. Blockchain Data Analysis: A Review of Status, Trends and Challenges[J]. Journal of Computer Research and Development, 2018, 55(9): 1853-1870.

[4] LUO W J, WEN S L, CHENG Y. Blockchain-based electronic health record sharing scheme[J]. Journal of Computer Applications, 2020, 40(1): 157-161.

[5] MAXWELL G, POELSTRA A, SEURIN Y, et al. Simple schnorr multi-signatures with applications to bitcoin[J]. Designs, Codes and Cryptography, 2019, 87(9): 2139-2164.

[6] DOERNER J, KONDI Y, LEE E, et al. Threshold ECDSA from ECDSA assumptions: the multiparty case[C]// 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 1051-1066.

[7] WANG R H, ZHANG L F, ZHOU H, et al. A Byzantine Fault Tolerance Raft Algorithm Combines with BLS Signature[J]. Journal of Applied Sciences, 2020, 38(1): 93-104.

[8] OUYANG L W, WANG S, YUAN Y, et al. Smart Contracts: Architecture and Research Progresses[J]. Acta Automatica Sinica, 2019, 45(3): 445-457.

[9] GUO S T, WANG R J, ZHANG F L. Summary of Principle and Application of Blockchain[J]. Computer Science, 2021, 48(2): 271-281.

[10] SHAMIR A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613.

[11] PEDERSEN T P. Non-interactive and information-theoretic secure verifiable secret sharing[C]// Annual International Cryptology Conference. Berlin: Springer, 1991: 129-140.

[12] AHMAT D, CHOROMA M, BISSYANDÉ T F. Multipath Key Exchange Scheme Based on the Diffie-Hellman Protocol and the Shamir Threshold[J]. IJ Network Security, 2019, 21(3): 418-427.

[13] GENNARO R, JARECKI S, KRAWCZYK H, et al. Secure distributed key generation for discrete-log based cryptosystems [C]// International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 1999: 295-310.

[14] NICK J, RUFFING T, SEURIN Y. MuSig2: Simple Two-Round Schnorr Multi-Signatures[R/OL]. Cryptology ePrint Archive, Report 2020/1261, 2020. https://eprint.iacr.org/2020/1261, 2020.

[15] YUAN C, XU M X, SI X M. Optimization Scheme of Consensus Algorithm Based on Aggregation Signature [J]. Computer Science, 2018, 45(2): 53-56.

[16] BONEH D, LYNN B, SHACHAM H. Short signatures from the Weil pairing[J]. Journal of Cryptology, 2004, 17(4): 297-319.

[17] HE H W, YAN A, CHEN Z H. Survey of Smart Contract Technology and Application Based on Blockchain[J]. Journal of Computer Research and Development, 2018, 55(11): 2452-2466.

[18] MENG X F, LIU L X. Blockchain and Data Governance[J]. National Science Foundation of China, 2020, 34(1): 12-17.

[19] CHOUDHURI A R, GREEN M, JAIN A, et al. Fairness in an unfair world: Fair multiparty computation from public bulletin boards[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 719-728.

[20] LIN L, HUANG X. A parallel algorithm for lagrange interpolation polynomial[J]. Journal-Xiamen University Natural Science, 2004, 43(5): 592-599.

LIU Feng, born in 1988, Ph. D, candidate, engineer, is a senior member of China Computer Federation. His main research interests include blockchain technology, data science and cognitive cross science.

QI Jia-yin, born in 1972, professor, Ph.D supervisor. Her main research interests include advanced technologies and management innovation.