BDPC 2024

CERTIFICATE

OF PARTICIPATION

This Certificate is Awarded to

Linli Mo

Xi'an Jiaotong-Liverpool University

A Secure, Flexible and Decentralized Data Sharing Scheme for InterPlanetary File System(C3048)

for the attendance and delivery of presentation on
2024 2nd International Conference on Big Data and Privacy Computing (BDPC 2024)
held during January 10-12, 2024 in Macau, China.

Sponsored by 澳門城市大學
Universidade da Cidade de Macau
City University of Macau

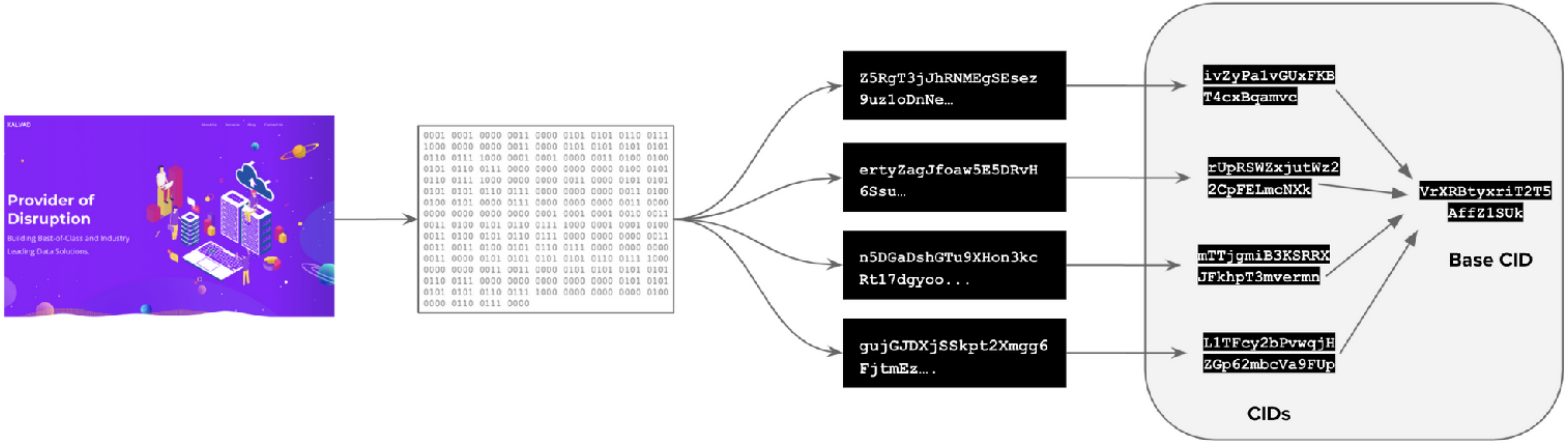
International Conference on Big Data and Privacy Computing ★ BDPC

Conference committee
BDPC 2024

- ### 1. 问题陈述
- ### 2. 相关工作
- ### 3. 设计目标
- ### 4. 系统概述
- ### 5. 关键功能
- ### 6. 实验结果
- ### 7. 结论

1. 问题陈述

- IPFS提供了P2P的文件存储，但是在存储私密数据的时候无法高效分享给别人。
- 我们需要一种安全、灵活的方式来在IPFS上共享隐私数据。



```
0001 0001 0000 0011 0000 0101 0101 0110 0111
1000 0000 0000 0011 0000 0101 0101 0101 0101
0110 0111 1000 0001 0001 0000 0011 0100 0100
0101 0110 0111 0000 0000 0000 0000 0100 0101
0110 0111 1000 0000 0000 0011 0000 0101 0101
0101 0101 0110 0111 0000 0000 0000 0011 0100
0100 0101 0000 0111 0000 0000 0000 0011 0000
0000 0000 0000 0000 0001 0001 0001 0010 0011
0011 0100 0101 0110 0111 1000 0001 0001 0100
0011 0100 0101 0110 0111 0000 0000 0000 0011
0011 0011 0100 0101 0110 0111 0000 0000 0000
0011 0000 0101 0101 0101 0101 0110 0111 1000
0000 0000 0011 0011 0000 0101 0101 0101 0101
0110 0111 0000 0000 0000 0000 0000 0101 0101
0101 0101 0110 0111 1000 0000 0000 0000 0100
0000 0110 0111 0000
```

Photo of Kalvad Page

Chunks

Digest

Merkle DAG

- Link: https://blog.kalvad.com/myths-about-ipfs/

2. 相关工作

- 1. 2018年，Steichen 使用 ACL 以太坊智能合约对 IPFS 进行了修改，以实现有效的文件共享。
- 2. 2020年，BATTAH 提出了一种基于去中心化区块链系统的 PRE 方案，用于多方访问加密的 IPFS 数据。
- 3. 2021年，Sharma 添加了 ABE 技术，构建了一个安全高效的基于区块链的云存储系统架构。

3. 设计目标

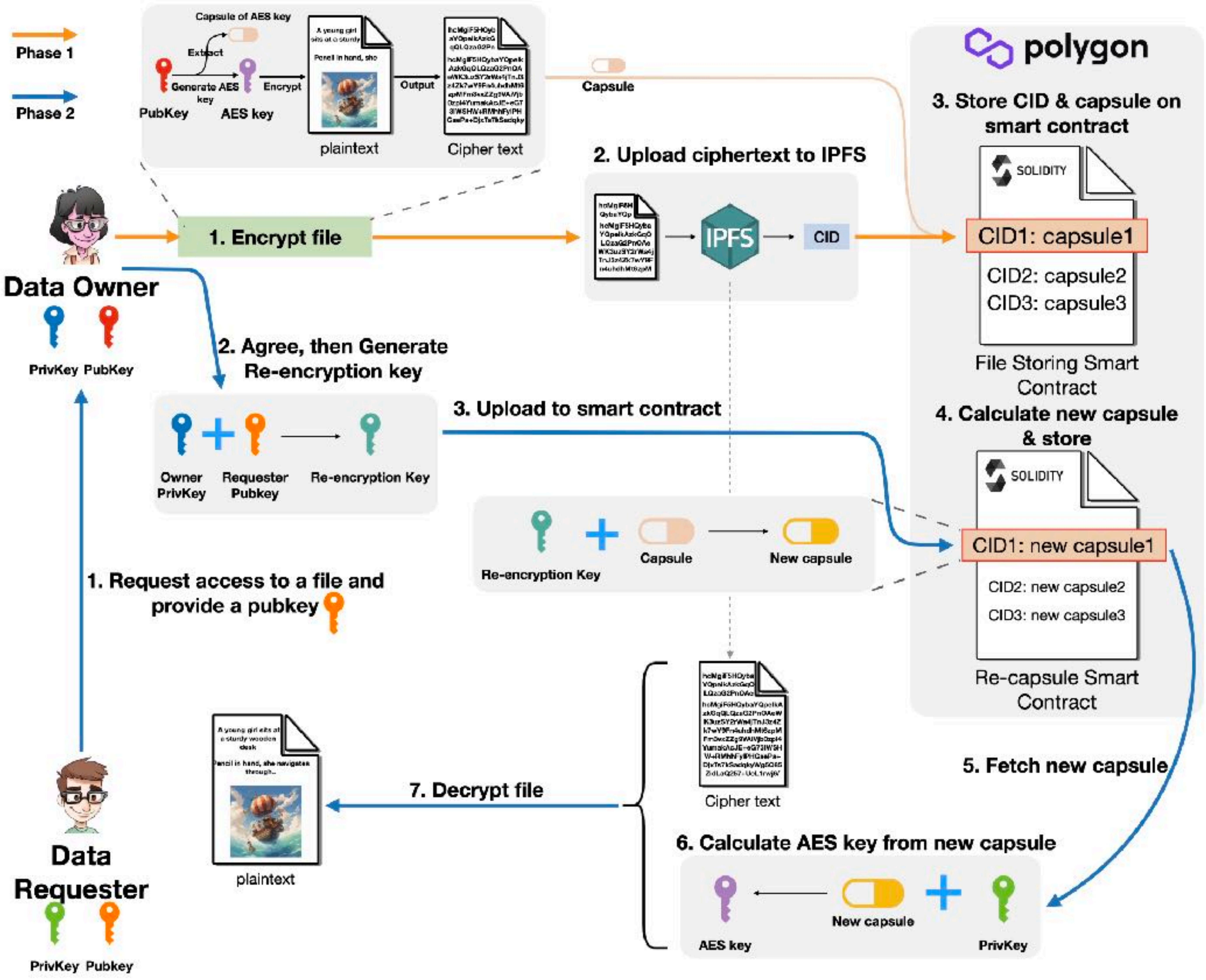
1.数据机密性：只有数据所有者和授权的数据请求者才能访问文件内容。

2.效率和成本效益：核心功能应在合理的时间内完成，并且花费的资源也要合理。

4. System Overview

黄色线段是数据加密阶段；

蓝色线段是数据分享阶段。



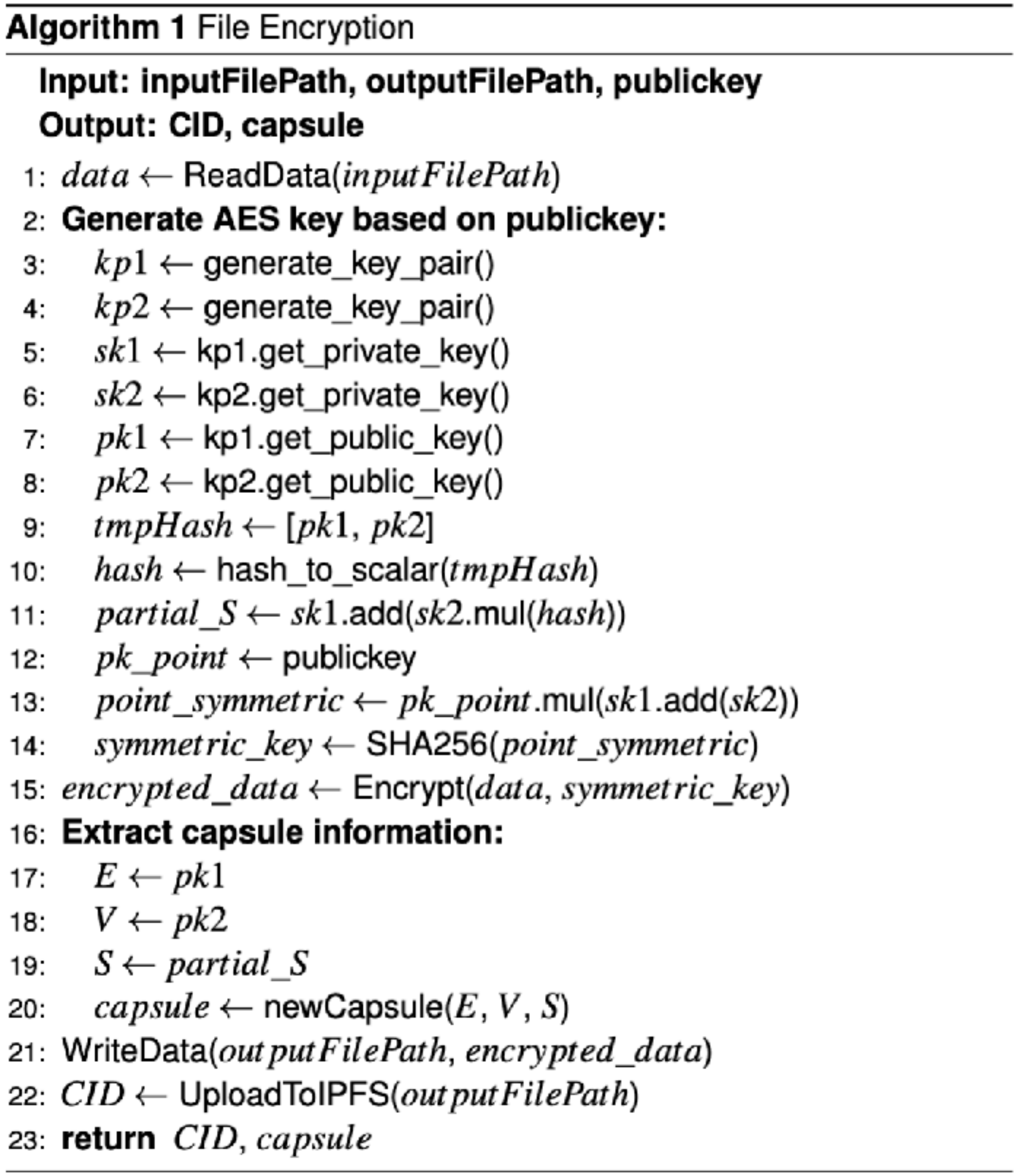
5.1 File Encryption Algorithm

主要逻辑如下：

1. 生成AES密钥

2. 使用AES加密文件

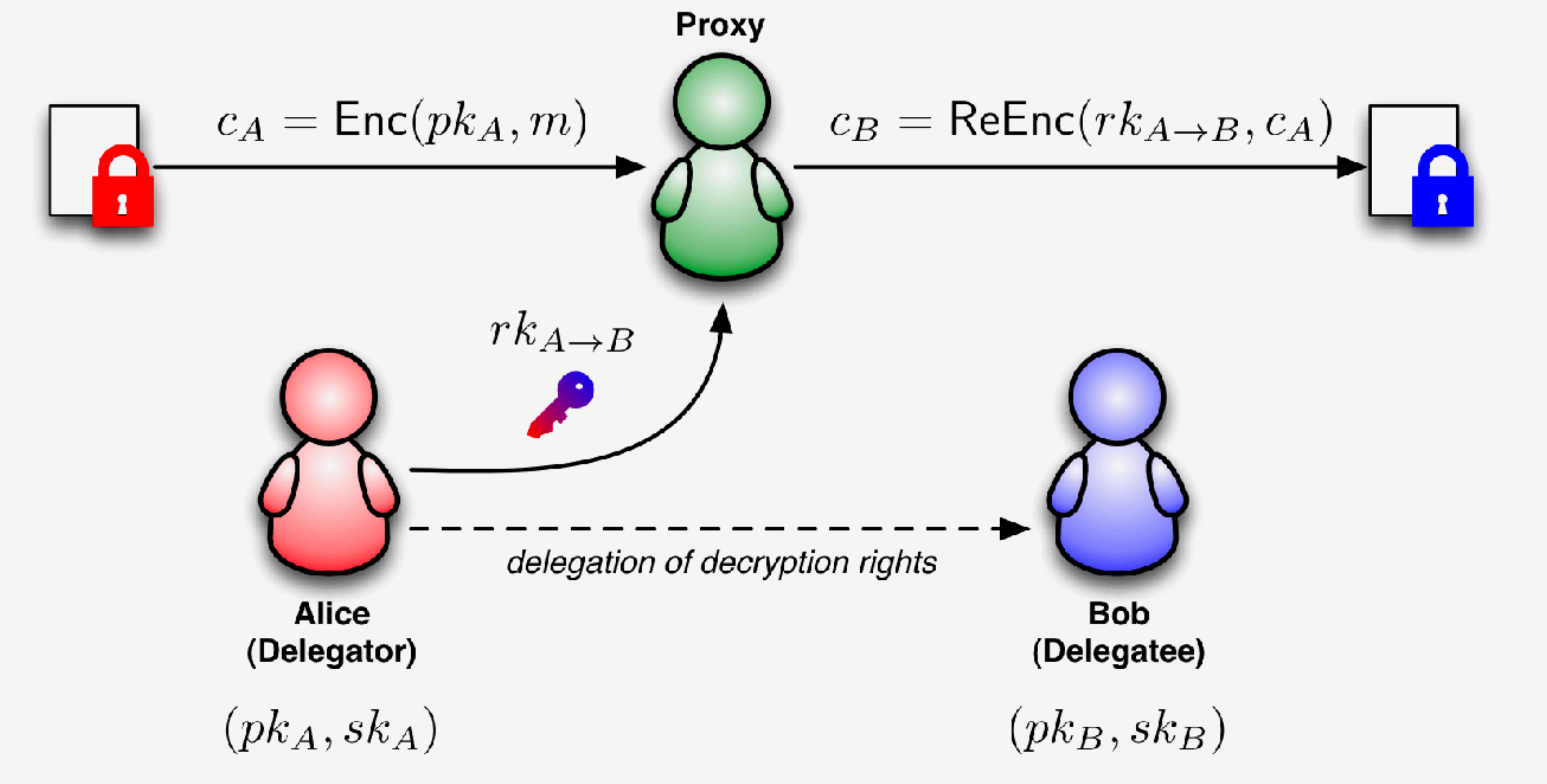
3. 获取AES密钥信息并且封装



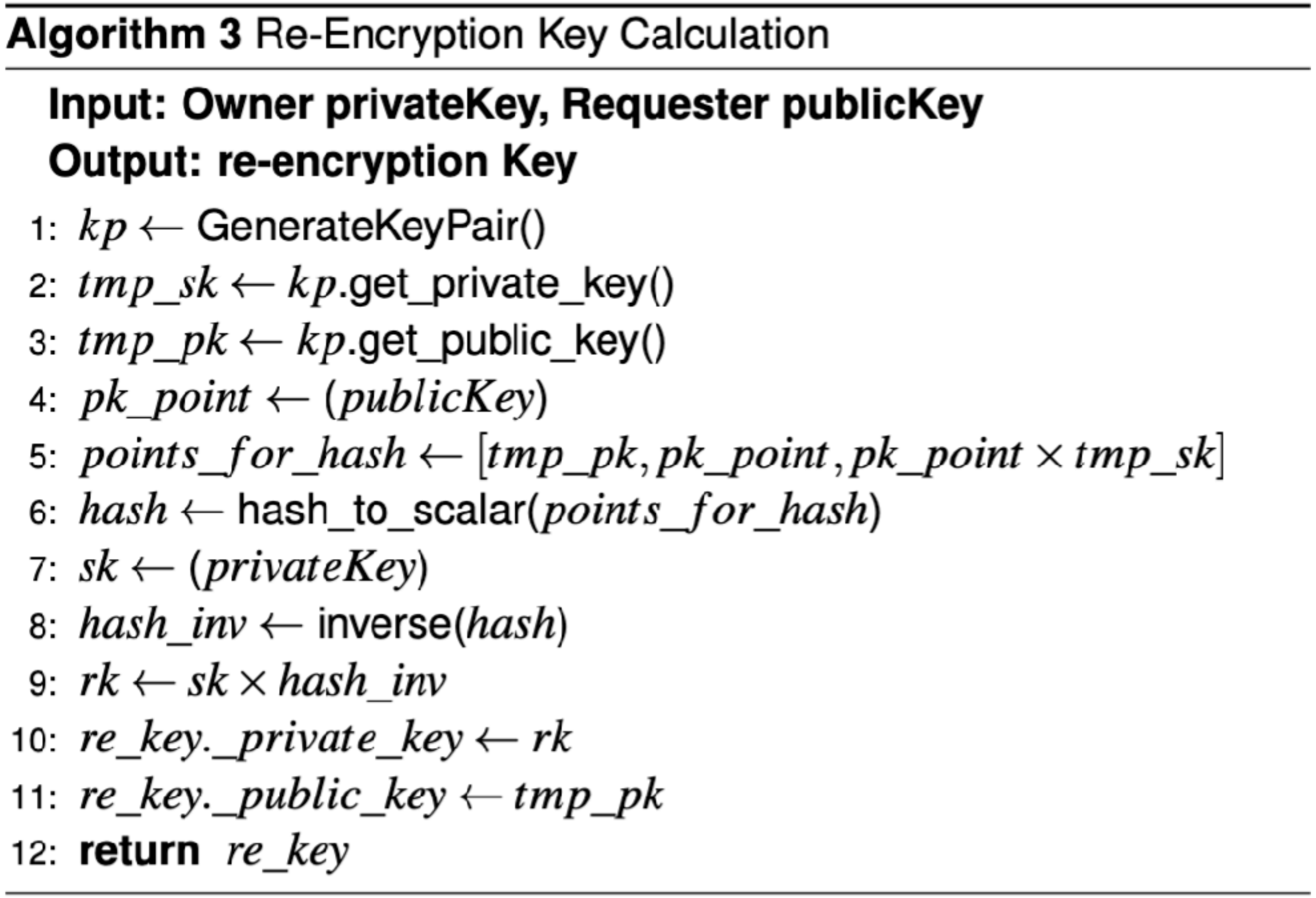
Algorithm 1 File Encryption

```
Input: inputFilePath, outputFilePath, publickey
Output: CID, capsule
$data \leftarrow$ ReadData($inputFilePath$)
Generate AES key based on publickey:
    $kp1 \leftarrow$ generate_key_pair()
    $kp2 \leftarrow$ generate_key_pair()
    $sk1 \leftarrow$ kp1.get_private_key()
    $sk2 \leftarrow$ kp2.get_private_key()
    $pk1 \leftarrow$ kp1.get_public_key()
    $pk2 \leftarrow$ kp2.get_public_key()
    $tmpHash \leftarrow [pk1, pk2]$
    $hash \leftarrow$ hash_to_scalar($tmpHash$)
    $partial\_S \leftarrow sk1$.add($sk2$.mul($hash$))
    $pk\_point \leftarrow$ publickey
    $point\_symmetric \leftarrow pk\_point$.mul($sk1$.add($sk2$))
    $symmetric\_key \leftarrow$ SHA256($point\_symmetric$)
$encrypted\_data \leftarrow$ Encrypt($data, symmetric\_key$)
Extract capsule information:
    $E \leftarrow pk1$
    $V \leftarrow pk2$
    $S \leftarrow partial\_S$
    $capsule \leftarrow$ newCapsule($E, V, S$)
WriteData($outputFilePath, encrypted\_data$)
$CID \leftarrow$ UploadToIPFS($outputFilePath$)
return $CID, capsule$
```

After encrypt file, we use Proxy Re-Encryption: modified Umbral algorithm without threshold


Proxy
$c_A = \mathsf{Enc}(pk_A, m)$
$c_B = \mathsf{ReEnc}(rk_{A\to B}, c_A)$
$rk_{A\to B}$
delegation of decryption rights
Alice
(Delegator)
(pk_A, sk_A)
Bob
(Delegatee)
(pk_B, sk_B)

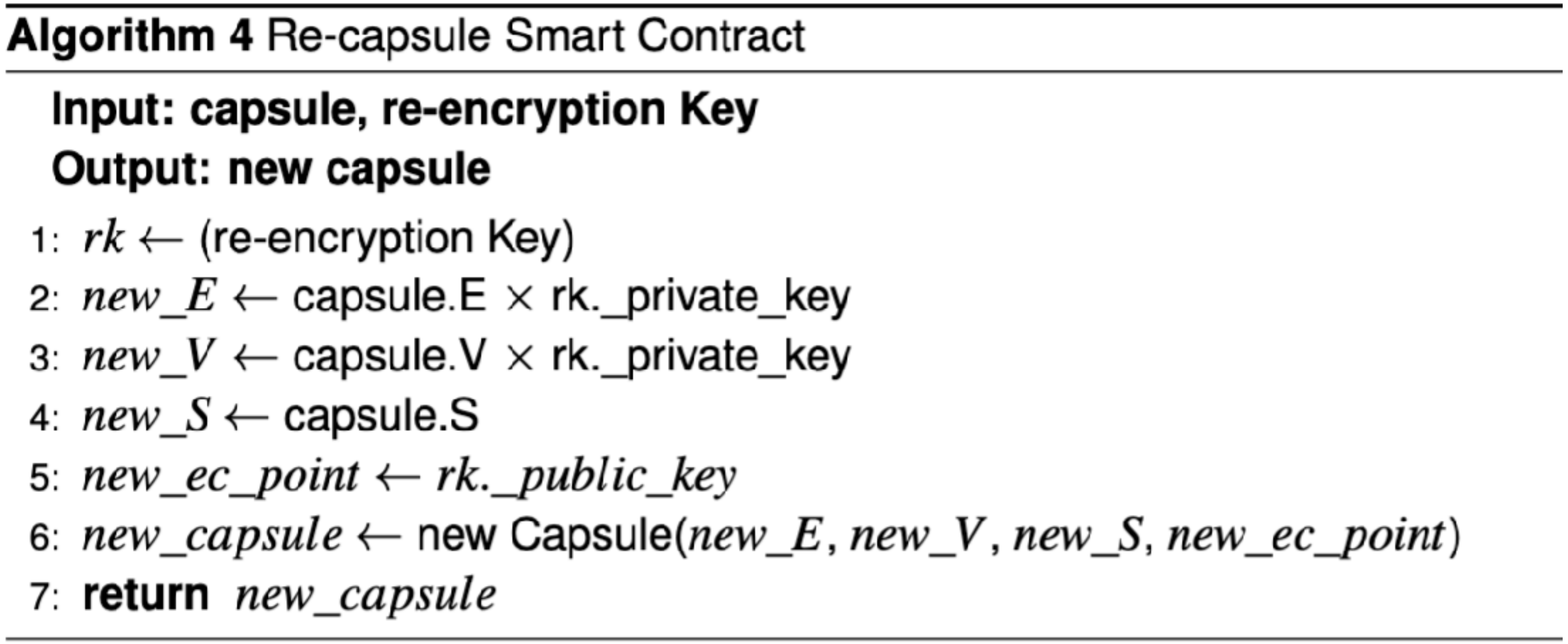
5.2 Re-Encryption Key Calculation



Algorithm 3 Re-Encryption Key Calculation

```
Input: Owner privateKey, Requester publicKey
Output: re-encryption Key
$kp \leftarrow$ GenerateKeyPair()
$tmp\_sk \leftarrow kp$.get_private_key()
$tmp\_pk \leftarrow kp$.get_public_key()
$pk\_point \leftarrow (publicKey)$
$points\_for\_hash \leftarrow [tmp\_pk, pk\_point, pk\_point \times tmp\_sk]$
$hash \leftarrow$ hash_to_scalar($points\_for\_hash$)
$sk \leftarrow (privateKey)$
$hash\_inv \leftarrow$ inverse($hash$)
$rk \leftarrow sk \times hash\_inv$
$re\_key.\_private\_key \leftarrow rk$
$re\_key.\_public\_key \leftarrow tmp\_pk$
return $re\_key$
```

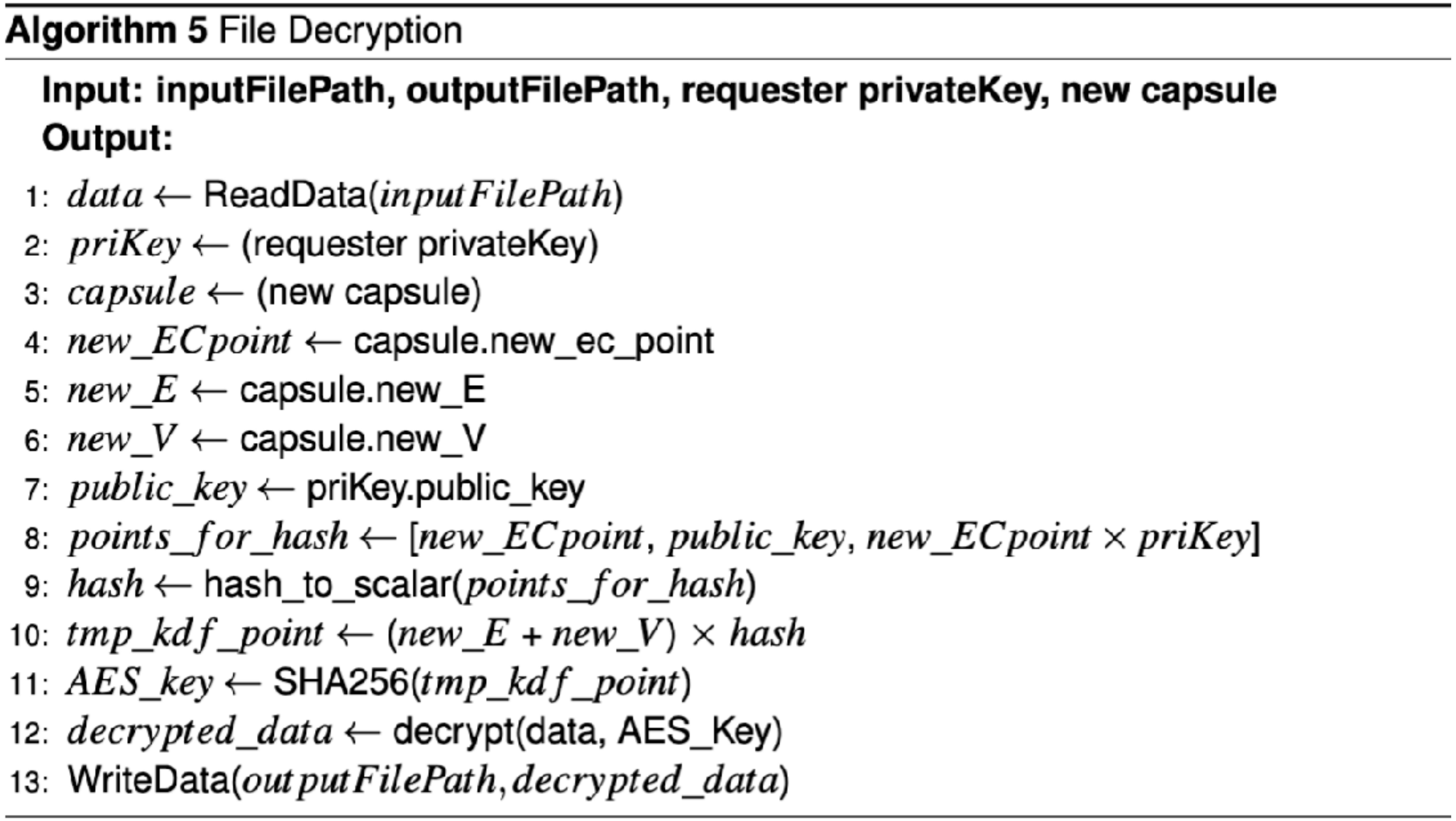
5.3 Re-capsule Smart Contract



Algorithm 4 Re-capsule Smart Contract

```
Input: capsule, re-encryption Key
Output: new capsule
$rk \leftarrow$ (re-encryption Key)
$new\_E \leftarrow$ capsule.E $\times$ rk._private_key
$new\_V \leftarrow$ capsule.V $\times$ rk._private_key
$new\_S \leftarrow$ capsule.S
$new\_ec\_point \leftarrow rk.\_public\_key$
$new\_capsule \leftarrow$ new Capsule($new\_E, new\_V, new\_S, new\_ec\_point$)
return $new\_capsule$
```

5.4 File Decryption Algorithm



Algorithm 5 File Decryption

```
Input: inputFilePath, outputFilePath, requester privateKey, new capsule
Output:
$data \leftarrow$ ReadData($inputFilePath$)
$priKey \leftarrow$ (requester privateKey)
$capsule \leftarrow$ (new capsule)
$new\_ECpoint \leftarrow$ capsule.new_ec_point
$new\_E \leftarrow$ capsule.new_E
$new\_V \leftarrow$ capsule.new_V
$public\_key \leftarrow$ priKey.public_key
$points\_for\_hash \leftarrow [new\_ECpoint, public\_key, new\_ECpoint \times priKey]$
$hash \leftarrow$ hash_to_scalar($points\_for\_hash$)
$tmp\_kdf\_point \leftarrow (new\_E + new\_V) \times hash$
$AES\_key \leftarrow$ SHA256($tmp\_kdf\_point$)
$decrypted\_data \leftarrow$ decrypt(data, AES_Key)
WriteData($outputFilePath, decrypted\_data$)
```

6. Experiment Results

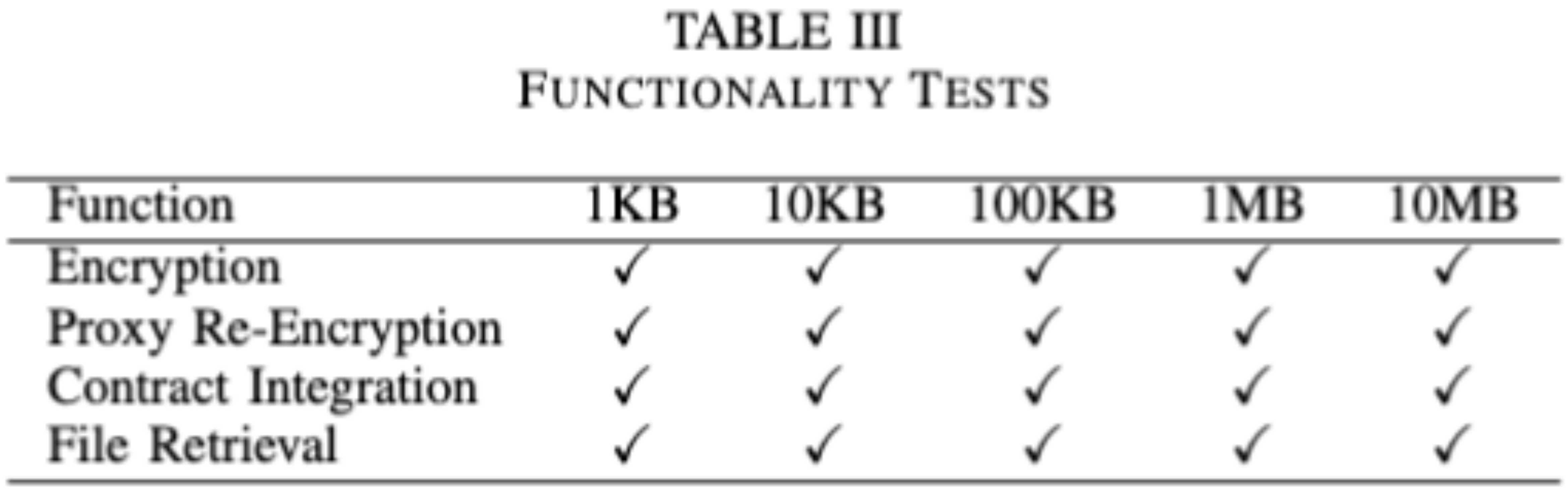
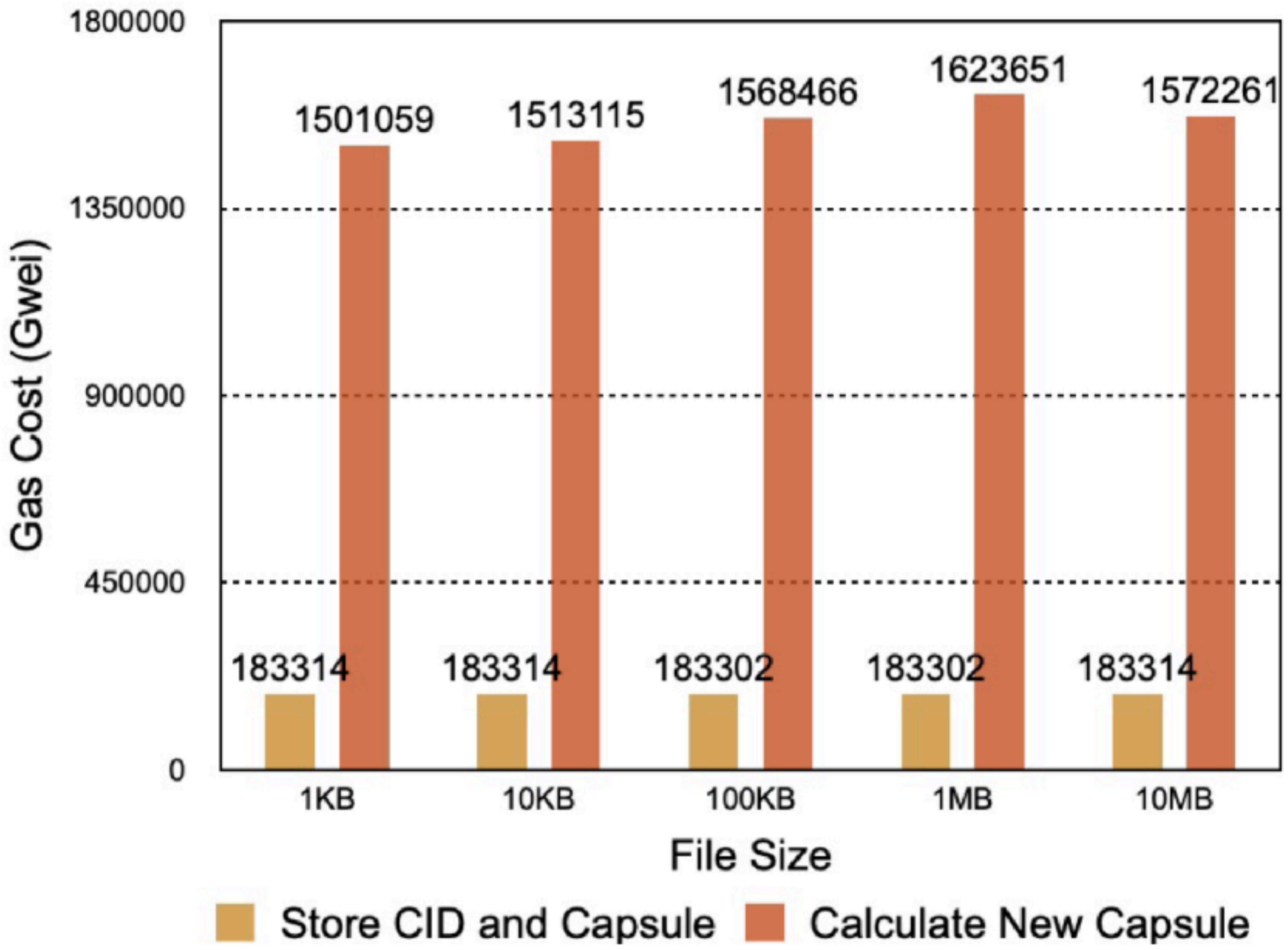


TABLE III
FUNCTIONALITY TESTS

Function	1KB	10KB	100KB	1MB	10MB
Encryption	✓	✓	✓	✓	✓
Proxy Re-Encryption	✓	✓	✓	✓	✓
Contract Integration	✓	✓	✓	✓	✓
File Retrieval	✓	✓	✓	✓	✓



8. Conclusion

- 我们的工作引入了一种新颖的解决方案，将区块链技术、代理重加密和智能合约结合起来，为IPFS提供了一种文件共享方案。