Hash Wires

姜舜天

2023 年 7 月 28 日

目录

1 HashWires

HashWires 的灵感来自 PayWord 协议，这是 Rivest 和 Shamir 在 1996 年提出的基于哈希链的微额支付协议。最初的想法非常简单，完全基于散列链计算。接 Project 6，使用 Project6 中的内容进行范围证明实际上只适用于小域，其对大范围（即 32 位或 64 位数）的性能并不实用。在这个 Project 中，我们对 HashWires 进行构造

1.1 MDP

为了扩展证明数的大小范围，避免出现 3997 无法证明 2999 的尴尬情况，我们使用 **Minimum Dominating Partitions(MDP)** 用于生成承诺，然后，根据要求证明的数字，Alice 将选择这些承诺之一：具有足够长的哈希多链来编码相关数字的承诺。MDP 生成的序列并非随机生成，MDP 产生最小的集合大小，满足上述属性，即能够证明到发行价值的任何范围。

核心代码如下：

Listing 1: MDP

```
def find_mdp_simple(value, base=10):
    exp = base
    mdp_list = [value]
    prev = value
    while exp < value:
        if (value + 1) % exp != 0:
            temp = int(value / exp) * exp - 1
            if prev != temp:
                mdp_list.append(temp)
                prev = temp
        exp *= base
    return mdp_list
```

1.2 生成 HashChain

核心代码如下：

Listing 2: Hash_Chains_generate

```
def hash_chain_generate(len, seed=None):
    if seed is None:
        seed = seed_generate()
    commitment = seed

    hash_chain = []
    for i in range(len):
        commitment = hashlib.sha256(commitment.encode('ascii')).hexdigest()
        hash_chain.append(commitment)
    return hash_chain, seed
```

1.3 生成多条 HashChains

核心代码如下：

Listing 3: multi_hash_chain_generate

```
def multi_hash_chain_generate(value, seeds=None):
    num = math.ceil(math.log10(value + 1))
    if not seeds:
        seeds = [seed_generate() for i in range(num)]
    multi_hash_chain = {i: hash_chain_generate(10, seeds[i])[0] for i in range(num)}
    return multi_hash_chain, seeds
```

1.4 HashChains 复用生成 HashWires

一个有趣的优化技巧是在 MDP 承诺之间共享链。实际上，这通过布线是直截了当的。简而言之，我们创建多个完整链，每个数字一个。然后，每个 MDP 承诺都连接到其相应的指数。核心代码如下：

Listing 4: HashWires_generate

```
# From hash multichain create the optimized hashwire
def hash_wire_commitment_generate(mdp_list, multi_hash_chain):
    # convert mdp_value to a list of digits
    digits = [int(d) for d in str(mdp_list)]
    # one mdp value can be shorter than the other
    diff = len(multi_hash_chain) - len(digits)

    return [multi_hash_chain[i + diff][digit] for i, digit in enumerate(digits)]
```

```python


# create list of hash wire for every mdp value
def commitment_hash_wire_generate(mdp_list, multi_hash_chain):
    return {value: hash_wire_commitment_generate(value,
        multi_hash_chain) for value in mdp_list}
```

2 具体实现

Listing 5: HashChain 类

```python
class HashChains:
    def __init__(self, int_value=None):
        self.seeds = []
        self.hash_chains = {}
        self.mdp = []
        self.commitments = []

        # set values when it's possible
        if int_value:
            self.create_hash_chains(int_value)
            self.create_mdp_list(int_value)
            self.create_commitments()

    def set_hash_chains(self, seeds=None, hash_chains=None):
        if (seeds and hash_chains) and \
                (len(seeds) == len(hash_chains)):
            self.seeds = seeds
            self.hash_chains = hash_chains
        elif seeds and not hash_chains:
            # update hashchain when seed is changed
            # mdp must be set before this is used!
            self.create_hash_chains(self.mdp[0], seeds)

    def create_hash_chains(self, int_value, seeds=None):
        if seeds:
            h, s = multi_hash_chain_generate(int_value, seeds)
        else:
            h, s = multi_hash_chain_generate(int_value)
        # set values
        self.set_hash_chains(s, h)

    def create_mdp_list(self, int_value):
        self.mdp = find_mdp_simple(int_value)

    def create_commitments(self):
        self.commitments = commitment_hash_wire_generate(self.mdp,
            self.hash_chains)
```

3 实现效果

3.1 测试代码

测试代码如下

Listing 6: testbench

```
chains = HashChains(17532)
print('MDP list is:',chains.mdp)
print('Seeds are:',chains.seeds)
for i in chains.commitments:
    print(i, chains.commitments[i])
```

3.2 执行结果

执行结果如下

Listing 7: testoutcome

```
MDP list is: [17532, 17529, 17499, 16999, 9999]
```

```
Seeds are: ['69
    f70cc08ede6c4876b990eb332266f54dca5aeabd29fec8313f0692359cbd84',
     '568
    fcaeaa9a2b0e9540888ca3e82f8d670510c15719524e468e9286b0b955b23',
    '
    eeda38a52c87558266181e0477f1214785b6772b0ade5ebe07f7aa3b7f1fb946
    ', '92
    bb29a3249b41e90e147df863f4d315719e7f93766da90af9b2ea691402ba87',
     '746
    a10ab32bd409f85738977845197ca5a4395e0292b12743b432da05680c412']
17532 ['8
    de8ce4ccfabfe916c5a65caea5ec51402823550d4b595282789151b6b140e75'
    , '
    e92a2d1d5051ebdceb29c5c413ba11b626b2175892d0d261811312feef7ed96c
    ', '7
    f6fdb136e31a965786607fef388e723f48ae4effa2820885d0d7c313789f581'
    , '16
    f8dc0ccf53c3c98d31861c313d5987ff4512d95efc36562964b63fd25d2116',
     '
    dfecb0892c13ab832fbb6b85378dadb042065e21aa4c2c6440167a1920d0656f
    ']
17529 ['8
    de8ce4ccfabfe916c5a65caea5ec51402823550d4b595282789151b6b140e75'
    , '
    e92a2d1d5051ebdceb29c5c413ba11b626b2175892d0d261811312feef7ed96c
    ', '7
    f6fdb136e31a965786607fef388e723f48ae4effa2820885d0d7c313789f581'
    , '6
    efe85462ff39d4629a131ba8e263aba2b819845177ee5283fd41a81a2b58e9b'
    , '
    f09f82bb8e66615a70be969adc6336cc71815cf0454868e3afea6fd6dad6f16e
    ']
17499 ['8
    de8ce4ccfabfe916c5a65caea5ec51402823550d4b595282789151b6b140e75'
    , '
    e92a2d1d5051ebdceb29c5c413ba11b626b2175892d0d261811312feef7ed96c
    ', '
    c2628717cf98f854ab852d02fc20099f91faf7e3595e668469d2e70ed50e53fa
    ', '
    cec475bc61b88419d38808a4efcff6dd33e2d5b15bd1a8c226ce4c9b746a44df
    ', '
    f09f82bb8e66615a70be969adc6336cc71815cf0454868e3afea6fd6dad6f16e
    ']
16999 ['8
    de8ce4ccfabfe916c5a65caea5ec51402823550d4b595282789151b6b140e75'
    , '
    e3b87538ac277d3797b416f99c8e44fd4c92ef47883b8ed7e4be02cd3071e8f9
    ', '3
    ce1f088106763bffb1dd54c6b2e0fdc0e054d0e76cfc09cf5c7ca022d3845e3'
    , '
    cec475bc61b88419d38808a4efcff6dd33e2d5b15bd1a8c226ce4c9b746a44df
    ', '
    f09f82bb8e66615a70be969adc6336cc71815cf0454868e3afea6fd6dad6f16e
    ']
9999 ['77
    b83f2a95aa1ebf0056bacfea697162dc869599855965a1ae97d0d9f996003e',
     '3
    ce1f088106763bffb1dd54c6b2e0fdc0e054d0e76cfc09cf5c7ca022d3845e3'
    , '
    cec475bc61b88419d38808a4efcff6dd33e2d5b15bd1a8c226ce4c9b746a44df
    ', '
    f09f82bb8e66615a70be969adc6336cc71815cf0454868e3afea6fd6dad6f16e
    ']
```

可以看到 MDP 测试如参考资料相同，HashWires 也正常生成，成功实现 HashWires

参考文献

[1] https://zkproof.org/2021/05/05/hashwires-range-proofs-from-hash-functions/