Key Exposure Problem of Chameleon Hashing

****

12-4-2024

Literature Review

Title	Date	Author(s)	Strategy
Chameleon Hashing Without Key Exposure	2004	Xiaofeng Chen, Fangguo Zhang, Kwangjo Kim	Alter private key with signature
On the Key Exposure Problem in Chameleon Hashes	2004	Giuseppe Ateniese Breno de Medeiros	Multi
Chameleon Hashes Without Key Exposure Based on Factoring	2007	Wei Gao,Xueli Wang,Dongqing Xie HNU	Factoring
Key-Exposure Free Chameleon Hashing and Signatures Based on Discrete Logarithm Systems	2009	Xiaofeng Chen, Fangguo Zhang, Haibo Tian, Baodian Wei, Kwangjo Kim	Discrete Logarithm
Identity-based chameleon hashing and signatures without key exposure	2014	Xiaofeng Chen, Fangguo Zhang, Willy Susilo, Haibo Tian, Jin Li, Kwangjo Kim	Identity-Based
Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain	2021	Chunhui Wua, Lishan Keb, Yusong Duc	-

The Initial Scheme

- The secret key is easy to obtain
- Weak non-transferability
- Weak non-repudiation

```
SK = random.randint(1, q)
PK = pow( g , x , p )

r1 = random.randint(1, q)
CH = pow( g , h , p ) * pow( PK , r1 , p ) % p
r2 = exgcd(SK,q)[0] * ( H(m1) - H(M2) + SK * r1 ) % q
```

Simplified code

——Chameleon Hashing and Signatures
Hugo Krawczyk,Tal Rabin 1997

- **System Parameters Generation** $\mathcal{PG}$**:** Let G be a Gap Diffie-Hellman group generated by g, whose order is a prime q. The system parameters are $SP = \{G, q, g\}$.
- **Key Generation** $\mathcal{KG}$ **:** Each user randomly chooses an integer $x \in Z_q^*$ as his private key, and publishes his public key $y = g^x$. The validity of y can be ensured by a certificate issued by a trusted third party.
- **Hashing Computation** $\mathcal{H}$**:** On input the public key y of a certain user. Randomly chooses an integer $a \in Z_q^*$, and computes (g^a, y^a). Our novel hash function is defined as

$$h = \text{Hash}(m, g^a, y^a) = \boxed{g^m y^a}$$

- **Collision Computation** $\mathcal{F}$**:** For any valid hash value h, the algorithm $\mathcal{F}$ can be used to compute a hash collision with the trapdoor information x

$$\mathcal{F}(x, h, m, g^a, y^a, m') = (g^{a'}, y^{a'}),$$

where $g^{a'} = g^a g^{x^{-1}(m-m')}$ and $y^{a'} = y^a g^{m-m'}$.

Note that

$$\boxed{\begin{aligned} \text{Hash}(m', g^{a'}, y^{a'}) &= g^{m'} y^{a'} \\ &= g^{m'} y^a g^{m-m'} \\ &= g^m y^a \\ &= \text{Hash}(m, g^a, y^a) \end{aligned}}$$

and $< g, y, g^{a'}, y^{a'} >$ is a valid Diffie-Hellman tuple. Therefore, the forgery is successful.

Motivation

One disadvantage of the initial chameleon signature scheme is that signature forgery results in the signer recovering the recipient's trapdoor information, *i.e.*, private key. Therefore, the signer can use this informa-

——Chameleon Hashing without Key Exposure

Xiaofeng Chen

Notice that if the recipient forges the signature, and two pairs (m, r) and (m', r') become known to the signer (during a dispute), the signer can recover the secret key x of the recipient from $h = \boxed{g^m y^r = g^{m'} y^{r'}, \text{ giving } x = \frac{m'-m}{r-r'}}$.

——On the Key Exposure Problem in Chameleon Hashes

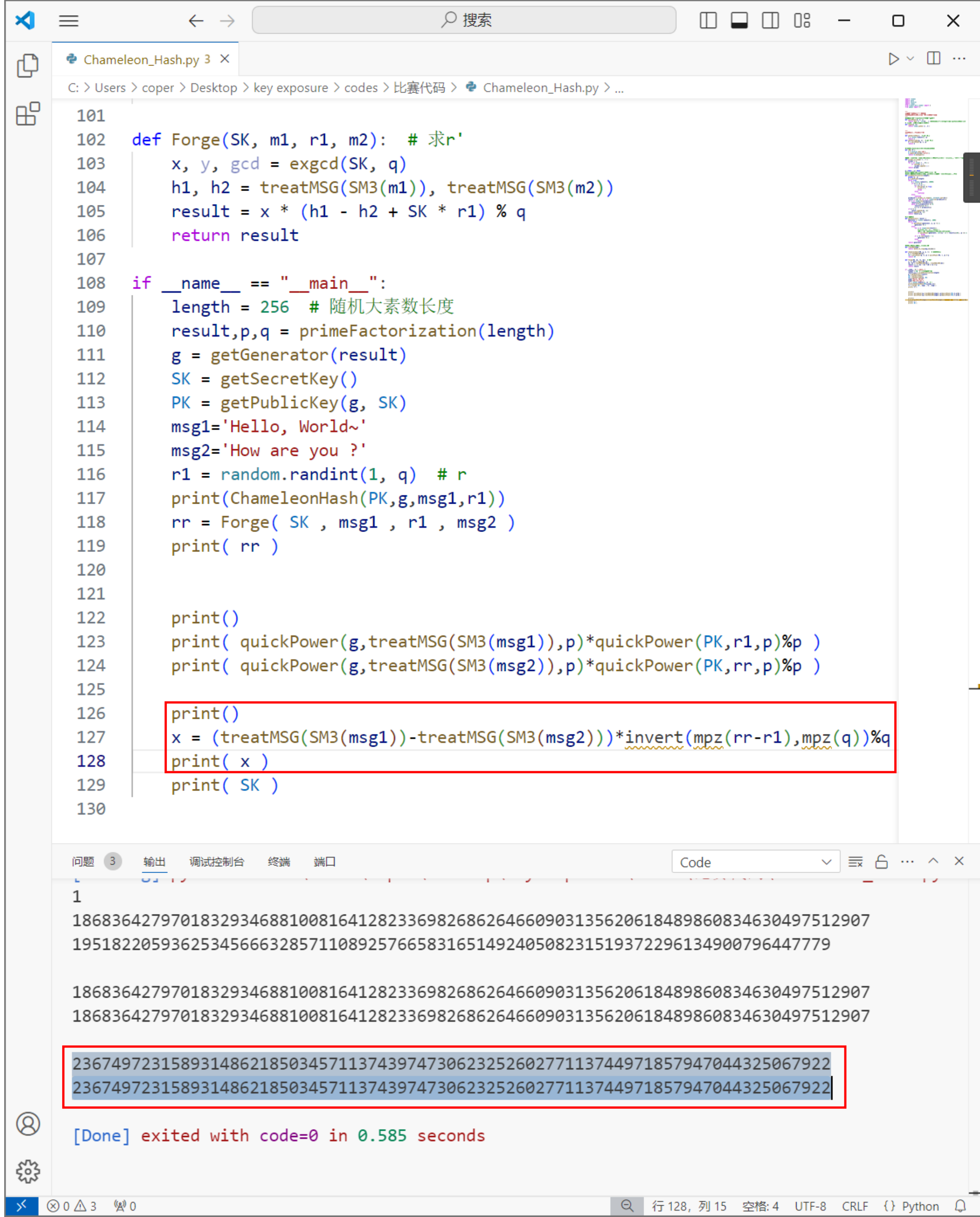
Giuseppe Ateniese

$$\text{已知 } g^m y^r = g^{m'} y^{r'}$$

$$\text{所以有 } g^m g^{x \cdot r} = g^{m'} y^{x \cdot r'}$$

$$\text{则有 } m + x \cdot r = m' + x \cdot r'$$

$$\text{移项得到 } x = \frac{m - m'}{r' - r}$$



1. Alter private key with signature

- Without key exposure
- Provide non-repudiation and tamper-proofing

cryptographic hash function, define $I = H(ID_S||ID_R||ID_T)$, where ID_S, ID_R, and ID_T denote the identity of signer, recipient, and transaction, respectively.

– **Signature Generation $\mathcal{SG}$:** Suppose the signed message is m. The signer S randomly chooses an integer $a \in Z_q^*$, and computes the chameleon hash function value $h = (g * I)^m y_R^a$, here y_R denotes the public key of the recipient R. Assume SIGN is any secure signature scheme based on the assumption that CDHP in G is intractable. The signature σ for the message m consists of

$$(m, I, g^a, y_R^a, \boxed{\mathrm{SIGN}_{x_S}(h)}).$$

Where x_S denotes the private key of the singer S.

– **Signature Verification $\mathcal{SV}$:** Given a signature σ, the recipient first verifies whether $< g, y_R, g^a, y_R^a >$ is a valid Diffie-Hellman tuple. If tuple is invalid, he rejects the signature; else, he then computes the chameleon hash value $h = (g * I)^m y_R^a$ and verifies the validity of $\mathrm{SIGN}_{x_S}(h)$ with the public key y_S of the signer.

Source: Xiaofeng Chen, Fangguo Zhang, Kwangjo Kim: Chameleon Hashing Without Key Exposure. Information Security.2004.87-98

2. Two Strategies

2.1. Single Trapdoor (Without Message Hiding)

- Security Dependency:

whether it is safe to sign the same message twice without redundancy

- Different:

place r in the exponent of y with e=H(m,r).

Key Generation: The scheme specifies a *safe* prime p of bitlength κ. This means that $p = 2q + 1$, where q is also prime, and a generator g of the subgroup of quadratic residues $\mathbf{Q}_p$ of $\mathbf{Z}_p^*$, i.e, g has order q. The recipient chooses as secret key x at random in $[1, q-1]$, and his public key is computed as $(g, y = g^x)$. Let $\mathcal{H}$ be a collision-resistant hash function, mapping arbitrary-length bitstrings to strings of fixed length τ: $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\tau$.

The Hash Scheme: To commit to a message m, it is sufficient to choose random values $(r, s) \in \mathbf{Z}_q \times \mathbf{Z}_q$, and compute:

$$e = \mathcal{H}(m, r); \text{ and } \mathbf{Hash}(m, r, s) = r - (y^e g^s \bmod p) \bmod q.$$

Collision Finding: Let C denote the output of the chameleon hash on input the triple (m, r, s). A collision (m', r', s') can be found by computing (m', r', s') such that:

$$e' = \mathcal{H}(m', r'); \text{ and } C = r' - (y^{e'} g^{s'} \bmod p) \bmod q.$$

First, the recipient chooses a random message m', a random value $k' \in [1, q-1]$, and computes $r' = C + (g^{k'} \bmod p) \bmod q$, $e' = \mathcal{H}(m', r')$, and $s' = k' - e'x \bmod q$. Notice that indeed:

$$r' - (y^{e'} g^{s'} \bmod p) \bmod q = C + (g^{k'} \bmod p) - (g^{xe'} g^{s'} \bmod p) \bmod q) = C.$$

Key Exposure Freeness and Collision-Resistance: The security of the scheme depends on whethe twice signing a message (without redundancy), using the above variant of Nyberg-Rueppel, is secure. This was proven in appendix A to [14], where the concept of twinning signature schemes is considered. The only difference from the scheme above is that we have substituted $e = \mathcal{H}(m, r)$ for r in the exponent of y. The only modification to the proof, which is formally the same, is that the probability of collisions is changed from finding collisions in the whole ring $\mathbf{Z}_q$ to finding them over the image of $\mathcal{H}(\cdot)$. Therefore, provided that this hash is collision-resistant, the conclusion of security is unchanged. Notice that we do not need to model the function $\mathcal{H}(\cdot)$ as a random oracle. Instead, the proof of security for the twin Nyberg-Rueppel works in the generic model of computation.

Source: Giuseppe Ateniese, Breno de Medeiros: On the Key Exposure Problem in Chameleon Hashes. Security in Communication Networks.2004.165-197

2. Two Strategies

2.2. Double Trapdoors (With Message Hiding)-Based on RSA

Hash Function: $H(\mathcal{L}, m, r) = J^{\mathcal{H}(m)} r^e \mod n$

Where $\mathcal{L}$ is a label, $B = J^d$, J depend on $\mathcal{L}$

已知 $J^{\mathcal{H}(m)} r^e = J^{\mathcal{H}(m')} r'^e$

所以有 $J^{\mathcal{H}(m)\cdot{-e}} r^{e\cdot{-e}} = J^{\mathcal{H}(m')\cdot{-e}} r'^{e\cdot{-e}}$

则有 $J^{\mathcal{H}(m)\cdot d} r = J^{\mathcal{H}(m')\cdot d} r'$

移项得到 $r' = rJ^{d\cdot\mathcal{H}(m) - \mathcal{H}(m')}$

即 $\boxed{r' = rB^{\mathcal{H}(m) - \mathcal{H}(m')}}$

$$\frac{r'}{r} = B^{\mathcal{H}(m) - \mathcal{H}(m')}$$

B is easy to be calculate if adversary have a pair of (m, r)

But the secret key that d remains safe

In this way everybody can forge a pair of new message,

which so called **Message Hiding**

Source: Giuseppe Ateniese, Breno de Medeiros: On the Key Exposure Problem in Chameleon Hashes. Security in Communication Networks.2004.165-197

3. Based on Factoring

3.1. Signature Scheme

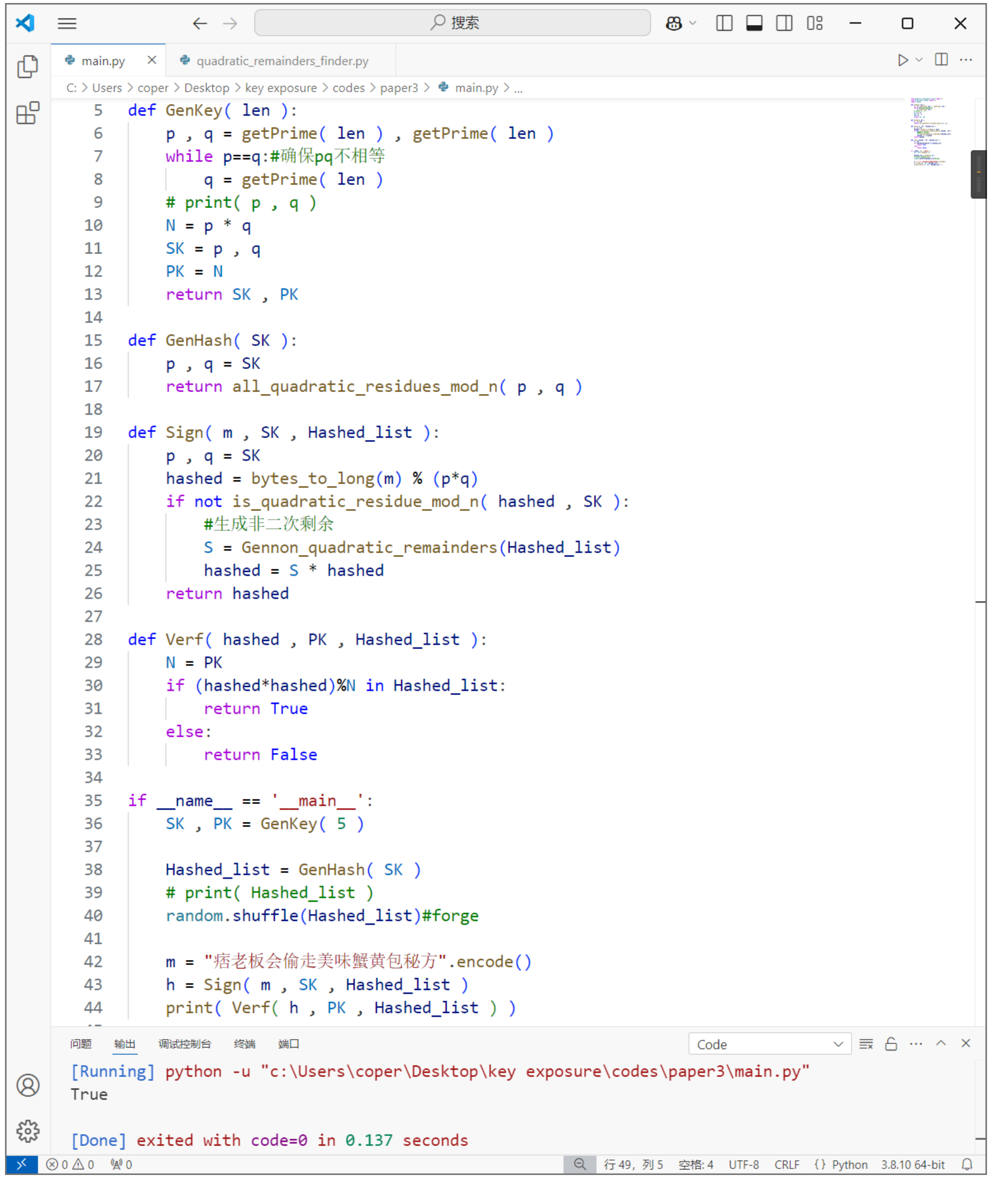
- **Sign.** On input the message $m \in \{0,1\}^*$, set the signature as:

$$\sigma \xleftarrow{R} |H(m)|^{\frac{1}{2}} \mod N$$

where if $H(m) \in QR_N$, $|H(m)| = H(m)$, else $|H(m)| = -H(m)$.

- **Ver.** On input σ, m, verify the following equality $\sigma^2 \equiv \pm H(m) \mod N$. In other words, if $\sigma^2 \equiv H(m) \mod N$ or $\sigma^2 \equiv -H(m) \mod N$, the signature is valid.

If one know the secret key (p,q), Quadratic and non-quadratic remainders is computable by CRT, or it's a hard problem.

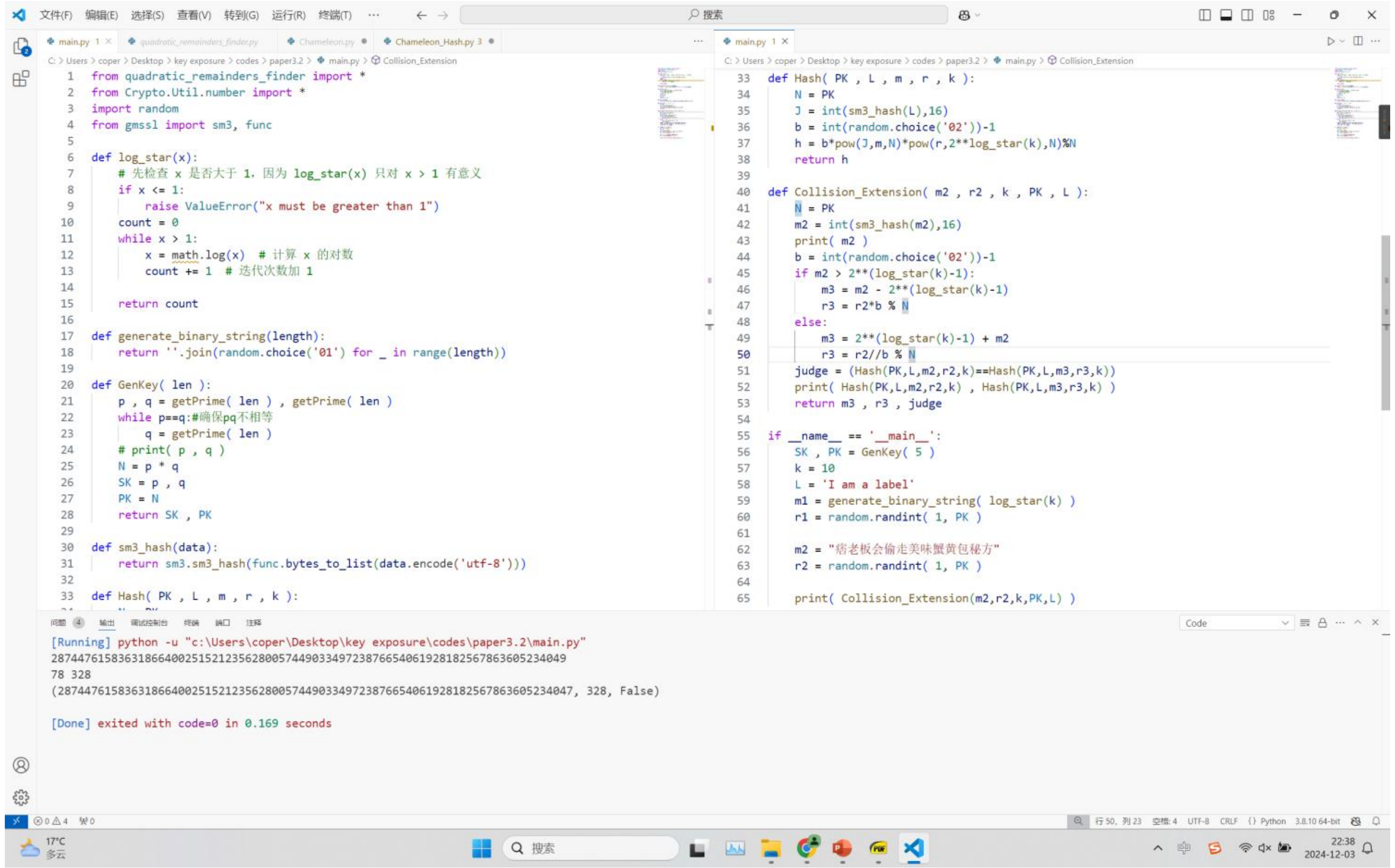


Source: Wei Gao, Xue-Li Wang, Dong-Qing Xie: Chameleon Hashes Without Key Exposure Based on Factoring. Journal of Computer Science and Technology.2007.109-113

3. Based on Factoring

3.2. Chameleon Hash

暂未跑通实验代码
有待后续继续研究



Source: Wei Gao, Xue-Li Wang, Dong-Qing Xie: Chameleon Hashes Without Key Exposure Based on Factoring. Journal of Computer Science and Technology.2007.109-113

4. Discrete Logarithm

r' is not a direct random number, but is generated through the random number a and a pair of $message$, and can be calculated by the equation

$$calculate\ H : H = g^a h^m$$
$$calculate\ r' : r' = (g^{a'}, y^{a'}) = (g^a h^{m-m'}, g^a h^{x(m-m')})$$

– **System Parameters Generation $\mathcal{PG}$:** Let $\mathbb{G}$ be a GDH group generated by g, whose order is a prime q. Let $H : \{0,1\}^* \to \mathbb{G}^*$ be a full-domain collision-resistant hash function. The system parameters are $SP = \{\mathbb{G}, q, g, H\}$.

– **Key Generation $\mathcal{KG}$:** Any user randomly chooses an integer $x \in_R \mathbb{Z}_q^*$ as his trapdoor key, and publishes his hash key $y = g^x$. The validity of y can be ensured by a certificate issued by a trusted certification authority.

– **Hashing Computation $\mathcal{H}$:** On input the hash key y, a customized identity I, let $h = H(y, I)$. Chooses a random integer $a \in_R \mathbb{Z}_q^*$, and computes $r = (g^a, y^a)$. Our proposed chameleon hash function is defined as

$$\mathcal{H} = \mathrm{Hash}(I, m, r) = g^a h^m.$$

– **Collision Computation $\mathcal{F}$:** For any valid hash value $\mathcal{H}$, the algorithm $\mathcal{F}$ can be used to compute a hash collision with the trapdoor key x as follows:

$$\mathcal{F}(\mathcal{H}, x, I, m, r, m') = r' = (g^{a'}, y^{a'}),$$

where $g^{a'} = g^a h^{m-m'}$ and $y^{a'} = y^a h^{x(m-m')}$.

Note that

$$\mathrm{Hash}(I, m', r') = g^{a'} h^{m'} = g^a h^{m-m'} h^{m'} = g^a h^m = \mathrm{Hash}(I, m, r)$$

Source: Xiaofeng Chen, Fangguo Zhang, Haibo Tian, Baodian Wei, Kwangjo Kim: Key-Exposure Free Chameleon Hashing and Signatures Based on Discrete Logarithm Systems.2009

4. Discrete Logarithm

——Simulation

最一开始按照论文逻辑写代码，一直跑不通

```python
def calc_c( m1 , a1 , m2 , SK , PK , g , q ):
    r2 = ( pow(g,a1,q)*pow(h,SM3(m1)-SM3(m2),q)%q , pow(y,a1,q)*pow(h,x*(SM3(m1)-SM3(m2)),q)%q )
    return r2

if __name__ == '__main__':
    q , g , I = init( 512 )
    SK = getSecretKey(q)
    PK = getPublicKey(g, SK, q)
    h = SM3(str(PK)+I)
    a1 = random.randint(1,q)
    r1 = ( pow(g,a1,q) , pow(PK,a1,q) )
    m1 = "hhhhh"
    m2 = "ggggg"
    a2 = random.randint(1,q)
    r2 = ( pow(g,a2,q) , pow(PK,a2,q) )
    # r2 = calc_c( m1 , a1 , m2 , SK , PK , g , q )
    print( Hash(h,m1,q,g,a1) )
    print( Hash(h,m2,q,g,a2) )
    print( Hash(h,m1,q,g,a1) == Hash(h,m2,q,g,a2) )
```

```
3490015744605821095222166784343852826080116158535713525308361250127347450085494241304946375453151010209025950251671
546795102425672888993141676329418087539
False
```

后来注意到a'是不可计算的，换了种写法就通了

paper4.py × | SM2_Signature.py | Chameleon_Hash.py 3 | OnlineOffline.py 1

C: > Users > coper > Desktop > key exposure > codes > paper4 > paper4.py > ...

```python
def init( len ):
    q = getPrime( len )
    g = fast_find_generator(q)
    I = "小明小绿小白"
    return q , g , I

def Hash( h , m , q , ga ):
    return ga*pow(h,SM3(m),q)%q


def calc_c( m1 , a1 , m2 , SK , PK , g , q , h ):
    x = SK ; y = PK
    r2 = ( pow(g,a1,q)*pow(h,SM3(m1)-SM3(m2),q)%q , pow(y,a1,q)*pow(h,x*(SM3(m1)-S
    return r2

if __name__ == '__main__':
    q , g , I = init( 512 )
    SK = getSecretKey(q)
    PK = getPublicKey(g, SK, q)
    h = SM3(str(PK)+I)
    a1 = random.randint(1,q)
    r1 = ( pow(g,a1,q) , pow(PK,a1,q) )
    m1 = "hhhhh"
    m2 = "ggggg"
    a2 = random.randint(1,q)
    # r2 = ( pow(g,a2,q) , pow(PK,a2,q) )
    r2 = calc_c( m1 , a1 , m2 , SK , PK , g , q , h )
    ga2 , ya2 = r2
    ga1 , ya1 = r1
    print( Hash(h,m1,q,ga1) )
    print( Hash(h,m2,q,ga2) )
    print( Hash(h,m1,q,ga1) == Hash(h,m2,q,ga2) )
```

```
[Running] python -u "c:\Users\coper\Desktop\key exposure\codes\paper4\paper4.py"
9033204062199135874655386338906634490202547772686166295295807399959181033947296974882328677563 7
2611019942113737758527361022228509898558614097215945477292 7
9033204062199135874655386338906634490202547772686166295295807399959181033947296974882328677563 7
2611019942113737758527361022228509898558614097215945477292 7
True

[Done] exited with code=0 in 0.2 seconds
```

行 95，列 30　空格: 4　UTF-8　CRLF　{} Python　3.8.10 64-bit

Source: Xiaofeng Chen, Fangguo Zhang, Haibo Tian, Baodian Wei, Kwangjo Kim: Key-Exposure Free Chameleon Hashing and Signatures Based on Discrete Logarithm Systems.2009

5. ID-Based

$$SetUp: (SK, PK) \leftarrow k$$ 生成公钥私钥

$$Extract: TK \leftarrow (SK, ID)$$ 生成与哈希密钥ID相关联的陷门密钥

$$Hash: Hash(ID, L, m, r) \leftarrow (PK, ID, L, m, r)$$

L为定制身份，m为消息，r为随机数，哈希结果不依赖于TK

$$Forge: r' = F(TK, ID, L, h, m, r, m')$$

Even if the hash function construction is not strong enough, it is only possible to leak ID. The hash result does not depend on TK, so there is no key leakage.

- **Setup:** *PKG runs this probabilistic polynomial-time algorithm to generate a pair of secret/public keys* (SK, PK) *defining the scheme. PKG publishes the system parameters* SP *including the public key* PK, *and keeps the secret key* SK *as the master key. The input to this algorithm is a security parameter* k.
- **Extract:** *A deterministic polynomial-time algorithm that, on input the master key* SK *and an identity string* ID, *outputs the trapdoor key* TK *associated to the hash key* ID.
- **Hash:** *A probabilistic polynomial-time algorithm that, on input the master public key* PK, *an identity string* ID, *a customized identity* L,[2] *a message* m, *and a random string* r,[3] *outputs the hash value* $h = \mathsf{Hash}(PK, ID, L, m, r$ *Note that* h *does not depend on* TK *and we denote* $h = \mathsf{Hash}(ID, L, m, r)$ *for simplicity throughout this paper.*
- **Forge:** *A deterministic polynomial-time algorithm* $\mathcal{F}$ *that, on input the trapdoor key* TK *associated to the identity string* ID, *a customized identity* L, *a hash value* h *of a message* m, *a random string* r, *and another message*

$m' \neq m$, *outputs a string* r' *that satisfies*

$$h = \mathsf{Hash}(ID, L, m, r) = \mathsf{Hash}(ID, L, m', r').$$

More precisely,

$$r' = \mathcal{F}(TK, ID, L, h, m, r, m').$$

Moreover, if r *is uniformly distributed in a finite space* $\mathcal{R}$, *then the distribution of* r' *is computationally indistinguishable from uniform in* $\mathcal{R}$.

Source: Xiaofeng Chen, Fangguo Zhang, Willy Susilo, Haibo Tian, Jin Li, Kwangjo Kim: Identity-based chameleon hashing and signatures without key exposure. Information Security and Privacy.2009.200–215

6. Quantum Resistant

- 后量子的，比较难懂，还没看完（悲）

Source: Chunhui Wu, Lishan Ke, Yusong Du: Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain. Information Sciences.2021.438-449

7. Summary and Outlook

- 3.2的实验代码
- 第六篇再花时间读一下，实在读不懂就搁置一下
- Survey的密钥泄露部分可以开写了
- 第二篇的第一个理论是否存在连分数攻击的安全风险