Blockchain Privacy Protection Scheme Based on Aggregate Signature

Kang Qiao
National Digital Switching System Engineering and Technological R&D Center
Zhengzhou, China
e-mail: 773441271@qq.com

Hongbo Tang
National Digital Switching System Engineering and Technological R&D Center
Zhengzhou, China

Wei You
National Digital Switching System Engineering and Technological R&D Center
Zhengzhou, China

Yu Zhao
National Digital Switching System Engineering and Technological R&D Center
Zhengzhou, China

***Abstract*—With the continuous development and wide application of blockchain technology, the problem of privacy leakage is becoming more and more prominent and must be fully taken seriously. This paper proposes a new blockchain signature scheme based on the aggregation signature scheme for the privacy protection of transaction addresses in blockchain. Compared with related schemes, the privacy protection scheme reduces the computational overhead of the signature and verification process, reduces the storage overhead of the blockchain, and improves the communication efficiency. The theoretical analysis results show that the scheme is safe and effective, and can protect the identity privacy of the receiver in the blockchain transaction.**

Keywords-blockchain; privacy protection; aggregate signature

I. INTRODUCTION

A blockchain is a concatenated transaction record (also called a block) that is concatenated and protected by cryptography [1]. Each block contains the cryptographic hash of the previous block, the corresponding timestamp, and the transaction data (usually represented by the hash value calculated by the Merkel tree algorithm) [2]. This design makes the block content difficult to tamper with. A distributed ledger concatenated with a blockchain allows both parties to effectively record transactions and permanently check the transaction. In 2008, Satoshi Nakamoto proposed the concept of "blockchain" in Bitcoin White Paper [3], and in 2009 founded the Bitcoin network to develop the first block, the "Creation Block" [4].

However, the global ledger that records transaction data in the blockchain is public in the network, and any attacker can obtain all transaction information, leaving the trader's privacy risk of disclosure. In view of the privacy leakage problem faced by the blockchain, some privacy protection schemes, have emerged, including the hybrid currency [5], the ring signature [6], and the zero-knowledge proof [7]. In order to break the interconnection between transactions, Maxwell proposed the idea of CoinJoin in 2013 [8], that is, several users execute the corresponding agreement to output the equivalent bitcoin transaction to the newly established address, thereby confusing the connection relationship in the transaction. There are many improvements and variants of the coinage agreement based on this idea [9-11]. ZeroCoin [12] only protects the privacy of the payer. On the basis of ZeroCash [13], the privacy protection content is extended to the payer, payment amount and payee to realize the full anonymous blockchain system, but the efficiency of the scheme is low. Ruffing and Moreno-Sanchez [14] combined the concepts of CT and CoinJoin in 2018 and proposed the ValueShuffle protocol for distributed covert transaction amounts. Yuan [15] et al. proposed a blockchain privacy protection scheme based on aggregated signatures, which aggregates multiple signature messages into one signature, which reduces the storage space of the signature and the bandwidth of the transmission network, and also reduces the workload of signature verification, but the scheme Based on the bilinear map construction, there is a lack of computational efficiency. This paper proposes an efficient short signature length aggregation signature scheme, which improves computational efficiency and storage efficiency, and achieves efficient blockchain privacy protection.

The remainder of this paper is organized as follows. This paper present the aggregate signature in section 2.Section3 introduces the core of signature scheme. Section 4 describes the security analysis of signature scheme. Section 5 summarizes contributions.

II. THE AGGREGATE SIGNATURE

Aggregate signature[16][17]is a digital signature variant that supports aggregation characteristics, that is, given n users $u_i \in U\left(1 \le i \le n\right)$, where U is the user set, for n messages, $m_i \in M\left(1 \le i \le n\right)$, where M is a set of messages. For n signatures, the generator of the aggregate signature (which may be different from u_i or untrusted) may aggregate

the n (single) signatures into a unique short signature σ, Given the aggregated signature, the original message m_i that participates in generating the identity u_i of the aggregated signer and its signature, the verifier can be assured that the user u_i has signed the message m_i [18].

$AS = \{Gen, Sign, Verify, AggS, AggV\}$ is a polynomial time algorithm hexad, which is described as follows:

$DS = (Gen, Sign, Verify)$ is a common signature scheme, also known as the base signature of the aggregate signature.

$AggS$: Aggregate signature generation algorithm, which can implement at least 3 sub-functions.

a) Implementing the ordinary signature function;

b) Implementing the aggregation of three vector groups, namely the message vector $(m_1,...,m_n)$, the user vector $(u_1,...,u_n)$, and aggregation function of individual signature vectors $(\sigma_1,...,\sigma_n)$;

c) Aggregation adds a new signature σ_{n+1}.

$AggV$:Aggregate signature verification algorithm, assuming that each user has a private key sk and a public key pk, if

$AggV(pk_1,...,pk_n,m_1,...,m_n,AggS(pk_1,...pk_n,Sign(sk_1,m_1),...,Sign(sk_n,m_n)))=1$, then output 1, otherwise output 0.

III. The Core of Signature Scheme

This paper proposes an efficient short signature length aggregate signature scheme, which consists of the following algorithms:

- ***System initialization algorithm***

  KGC chooses a safety parameter $k \in Z^+$ that satisfies $p > 2^k$. $\mathbb{F}_p$ is an elliptic curve cycle group whose order is prime p, and G is a generator of the group $\mathbb{F}_p$. KGC selects a random number $s \in_R Z_q^*$ as the system master key and calculates the system public key:

  $$P_{Pub} = sG$$

  Then pick the anti-collision secure hash function: the cryptographic hash function $H_s : \{0,1\}^* \to \mathbb{F}_p$, the determined hash function $H : E(\mathbb{F}_P) \to E(\mathbb{F}_P)$. Finally, expose the system parameter $Params = \langle k, p, G, \mathbb{F}_p, P_{Pub}, H_s, H \rangle$ and secretly save s.

- ***Secret value generation algorithm***

  Define an aggregated user collectio $U_1 \subseteq U, U_1 = \{u_1, u_2, ..., u_k\}$. Each user $u_i \in U_1$ randomly selects the real numbe $a_i \in [1, l-1], b_i \in [1, l-1] (a_i \neq b_i)$ as its key value.

- ***Private key generation algorithm***

  Calculate $A_i = a_i G, B_i = b_i G$, then the private key of the user's u_i is $SK_i = (a_i, b_i)$.

- ***Public key generation algorithm***

  Calculate $A_i = a_i G, B_i = b_i G$, then the public key of the user's u_i is $PK_i = (A_i, B_i)$.

- ***Signature algorithm***

  Given a transaction message m_i, the user u_i randomly selects the secret value $x_i \xleftarrow{R} \mathbb{Z}_p$ to calculate the shared information.

  $$V_i = x_i G$$

  And send V_i to the other $k-1$ users $u_j (1 \le j \le k, j \neq i)$, when u_i receives the information V_j shared by other $k-1$ users u_j, calculate

  $$V = \sum_{i=1}^{k} V_i$$
  $$h_i = H(m_i, V)$$
  $$S_i = x_i + (a_i + b_i) h_i$$

  Then the obtained signature $\sigma_i = \langle V, V_i, S_i \rangle$ is the signature of the user u_i on the transaction message m_i.

- ***Single verification algorithm***

  Given the signature $\sigma_i = \langle V, V_i, S_i \rangle$ with the user u_i about the message m_i, the verifier calculates

  $$V' = \sum_{i=1}^{k} V_i$$
  $$h_i^1 = H(A_i, B_i)$$
  $$h_i = H(m_i, V)$$

  Verification equation

  $$V' = V$$
  $$S_i G = V_i + h_i (A_i + B_i)$$

  If it is established, the certifier receives the signature and rejects if it is not established.

- ***Aggregation algorithm***

  The aggregator aggregates a different set of signatures to form an aggregate signature.

  a) Define the aggregate user set $U_1 \subseteq U, U_1 = \{u_1, u_2, ..., u_k\}$, the public key corresponding to each user $u_i \in U_1$ is $PK_i = (A_i, B_i)$, and the corresponding message signature pair $\{(m_1, \sigma_1 = (V, V_1, S_1)), ..., (m_n, \sigma_n = (V, V_n, S_n))\}$

  b) Then the aggregator U_{AggS} calculates

  $$S = \sum_{i=1}^{k} S_i$$

  Output the aggregate signature $\sigma = \langle V, S \rangle$ of

message $m_1, m_2, ..., m_n$.

- ***Aggregation verification algorithm***

  Given the system disclosure parameters, the message and public key $(m_i, PK_i)(1 \le i \le k)$, and the aggregate signature $\sigma = \langle V, S \rangle$, the signature verifier U_{Ver} performs the following steps to verify the aggregate signature.

  a) For $i = 1, 2, ..., k$, calculate $h_i = H(m_i, V)$

  b) For the aggregate signature $\sigma = \langle V, S \rangle$, verify the equation

$$SG = V + \sum_{i=1}^{k} h_i (A_i + B_i)$$

  If the equation is true, the output is **True,** indicating that $\sigma = \langle V, S \rangle$ is the aggregate signature of U_{AggS} for $(m_i, PK_i)(1 \le i \le k)$; if the equation is not true, **False** is output.

IV. Security Analysis of Signature Scheme

In this section, under the random prediction model, the unforgeability of the scheme is proved based on the discrete logarithm problem.

Lemma 1: Under the random prediction model, if there is a type A_1 adversary A_1, it can execute at most q_s signature inquiry, q_k partial partial key generation inquiry and q_{sk} private key generation inquiry, which can be ignored in time t. The probability ε breaks through the scheme of this paper, then there is such an algorithm C that can use the algorithm to succeed in the polynomial time with probability $Adv(C) \ge \left(1 - \frac{q_k}{q_C}\right)\left(1 - \frac{q_{sk}}{q_C}\right)\frac{\varepsilon}{ke(q_s + k)}$ (where q_C is the total number of queries, n is the number of users of the aggregated signature, and e is the natural log base) Solve the discrete logarithm problem.

Proof: Given a discrete logarithm problem instance (G, bG), where $b \in Z_q^*$ is unknown. C uses A_1 as a subroutine and acts as a challenger, then Challenger C interacts with adversary A_1. The goal of C is to solve the discrete logarithm problem. The following describes in detail how C uses A_1 to calculate b.

- ***System initialization phase:*** Challenger C runs the Setup algorithm to generate system parameters $Params = \langle k, p, G, \mathbb{F}_p, P_{Pub}, H_s, H \rangle$, let $P_{Pub} = bG$, and send Params to A_1.
- ***Interrogation phase:*** In order to avoid conflicts, the Challenger C maintenance list $L_1, L_K, L_{SK}, L_{PK}, L_S$ is used to track the oracle H, partial key generation, private key generation, public key generation and signature inquiry, respectively, and the list is initially empty. Adversary A_1 can adaptively initiate the following query to C:
- H ***query:*** When the enemy A_1 asks the oracle H for $H(A_i, B_i)$, the challenger C performs the following operations:

  a) Challenger C first queries list L_1. If there is a corresponding tuple $\langle A_i, B_i, h \rangle$ in list L_1, C returns h to A_1;

  b) If there is no corresponding tuple in the list L_1, C randomly selects $h \in Z_q^*$, so that there is no corresponding tuple $\langle *, *, h \rangle$ in L_1, and the corresponding tuples $\langle A_i, B_i, h \rangle$ to L_1 are added, and h is returned to A_1.

- ***Public key generation query:*** Challenger C has a list L_{PK}, which is initially empty. When adversary A_1 makes a public key generation query to user u_i, Challenger C performs the following operations:

  a) Challenger C first queries list L_{PK}. If there is a corresponding tuple $\langle u_i, A_i, B_i \rangle$ in list L_{PK}, C returns $PK_i = (A_i, B_i)$ to A_1;

  b) If there is no corresponding tuple in the list L_{PK}, C randomly selects the real number $a_i \in [1, l-1]$, calculates $A_i = a_i G$, performs partial key generation inquiry through u_i and A_i to know the tuple $\langle u_i, b_i, B_i \rangle$, and then challenger C adds $\langle u_i, A_i, B_i \rangle$ to the list L_{PK} and returns $PK_i = (A_i, B_i)$ gives the opponent A_1 and adds $\langle u_i, a_i, b_i \rangle$ to the list L_{SK}.

- ***The private key generation query:*** Challenger C has a list L_{SK}, which is initially empty. When the adversary A_1 makes a private key generation inquiry to the user u_i, the challenger C performs the following operations:

  a) Challenger C first queries list L_{SK}. If there is a corresponding tuple $\langle u_i, a_i, b_i \rangle$ in list L_{SK}, C returns $SK_i = (a_i, b_i)$ to A_1;

  b) If there is no corresponding tuple in the list L_{SK}, C randomly selects the real number $a_i \in [1, l-1]$,

calculates $A_i = a_i G$, performs partial key generation inquiry through $A_i = a_i G$ and A_i to know the tuple $\langle u_i, b_i, B_i \rangle$, and then challenger C adds $\langle u_i, a_i, b_i \rangle$ to the list L_{SK} and returns $SK_i = (a_i, b_i)$ to the enemy A_1 and will add $\langle u_i, A_i, B_i \rangle$ to the list L_{PK} .

- ***Partial key generation query:*** Challenger C has a list L_K , initially empty. When Adversary A_1 makes a partial key generation query for User u_i and Public Parameter A_i , Challenger C does the following:
  a) Challenger C first queries list L_K . If there is a corresponding tuple $\langle u_i, b_i, B_i \rangle$ in list L_K ,then C returns (b_i, B_i) to A_1 ;
  b) If there is no corresponding tuple in list L_K , if $u_i \neq u_j$, then C randomly select $h \in Z_q^*$, $b_i \in [1, l-1]$, calculate $B_i = b_i G - P_{Pub} h$, then challenger C adds $\langle u_i, b_i, B_i \rangle$ to list L_K , and returns (b_i, B_i) to adversary A_1 . If there is no corresponding tuple, then add $\langle u_i, A_i, B_i, h \rangle$ to L_1 ; if $u_i = u_j$, then C randomly selects $h \in Z_q^*$, $b_i \in [1, l-1]$, and $B_j = r_{know} G$ ($r_{know} \in Z_q^*$ is C known random number), then challenger C adds $\langle u_j, b_j, B_j \rangle$ to list L_K , and returns (b_j, B_j) to adversary A_1 , if the corresponding tuple does not exist, add $\langle u_i, A_i, B_i, h \rangle$ to L_1 .
- ***Public key replacement query:*** When Challenger C receives a request from the adversary A_1 for the user u_i 's public key replacement, a new public key $PK_i^{'} = (A_i^{'}, B_i^{'})$ can be selected to replace the original public key PK_i .
- ***Signature query:*** Challenger C has a list L_S , which is initially empty. When the enemy A_1 signs the user u_i to generate an inquiry, Challenger C performs the following operations:
  a) If there is no corresponding tuple in list L_S , if $u_i = u_j$, then challenger C gives up and ends the simulation;
  b) If $u_i \neq u_j$, then C randomly selects $x_i \in Z_q^*$, calculates $V_i = x_i G$, $V = \sum_{i=1}^{k} V_i$, $h_i = H(m_i, V)$ and $S_i = x_i + (a_i + b_i) h_i$, and generates signature $\sigma_i = \langle V, V_i, S_i \rangle$ to return to adversary A_1 .
- ***Aggregate signature query:*** When Adversa A_1 does an aggregate signature query, Challenger C does the following:
  a) For all $u_i (1 \leq i \leq k)$ there are $u_i \neq u_j$, then challenger C randomly selects $x_i \in Z_q^*$ for each user $u_i (1 \leq i \leq k)$, calculates $V_i = x_i G$, $V = \sum_{i=1}^{k} V_i$, $h_i = H(m_i, V)$, and $S_i = x_i + (a_i + b_i) h_i$, and then calculates $S = \sum_{i=1}^{k} S_i$ to generate an aggregate signature for $\sigma = \langle V, S \rangle$ to return to adversary A_1 ;
  b) Otherwise, Challenger C gives up and ends the simulation.
- ***Signature verification query:*** When the adversary A_1 makes a signature verification query, the challenger C first queries whether there is a u_i corresponding tuple in the list L_{PK} :
  a) If there is a u_i corresponding tuple in L_{PK} ,and $u_i \neq u_j$, then calculate $h_i^1 = H(A_i, B_i)$ and $h_i = H(m_i, V)$ to verify whether the equation $S_i G = V_i + h_i (A_i + B_i)$ is true: if it is established, the challenger C returns True to the adversary A_1 ; otherwise, returns Flase to the adversary A_1 ;
  b) Otherwise, Challenger C gives up and ends the simulation.
- ***Forgery stage:*** After the appeal is queried, the adversary A_1 outputs a quaternion (m^*, u^*, V^*, S^*) , where $m^* = (m_1^*, m_2^*, ..., m_k^*)$, $u^* = (u_1^*, u_2^*, ..., u_k^*)$, (V^*, S^*) represent the aggregated signatures of the k signatures $\sigma_i^* = \langle V, V_i^*, S_i^* \rangle$ generated by the k users u_i^* for the k different messages m_i^* .
  a) If there is $u_i \neq u_j$ for all $u_i \neq u_j$, then Challenger C gives up and ends the simulation;
  b) Otherwise, the challenger C queries the u_i corresponding record value in the list L_1, L_{SK}, L_{PK} , and verifies whether the equation $SG = V + \sum_{i=1}^{k} h_i (A_i + B_i)$ holds: if the equation is established, the challenger C outputs $b = (h_i^1 h_i)^{-1} \left\{ S - \sum_{i=1, i \neq j}^{n} \left[a_i + h_i (x_i + y_i) - a_i - h_i (x_i + r_{know}) \right] \right\}$ as an effective solution to the discrete logarithm problem; otherwise, Challenger C failed to solve

the discrete logarithm problem.

Challenger C The instance probability of solving the discrete logarithm problem can be transformed into the following three events:

Event ε_1 : At least one $u_f\left(f \in [1,k]\right)$ does not perform partial key generation inquiry and private key generation inquiry.

Event ε_2 : Challenger C did not terminate when signing the inquiry.

Event ε_3 : Challenger C did not terminate during the challenge phase. That is, the enemy A_1 falsifies an aggregate signature in the challenge phase, and the probability that the challenger C does not terminate in the challenge phase is $\Pr[\varepsilon_3] = \delta$.

Throughout the simulation, if the above events occur, then Challenger C wins the game, then the probability that C does not terminate is $\Pr[\varepsilon_1 \wedge \varepsilon_2 \wedge \varepsilon_3] = \Pr[\varepsilon_1]\Pr[\varepsilon_2 \mid \varepsilon_1]\cdot\Pr[\varepsilon_3 \mid \varepsilon_1 \wedge \varepsilon_2]$, where $\Pr[\varepsilon_1] \geq \frac{1}{k}\left(1-\frac{q_k}{q_C}\right)\left(1-\frac{q_{sk}}{q_C}\right)$, $\Pr[\varepsilon_2 \mid \varepsilon_1] \geq (1-\sigma)^{q_s}$, $\Pr[\varepsilon_3 \mid \varepsilon_1 \wedge \varepsilon_2] \geq \delta$. Since C has a probability of selecting user u_i as $\delta \in \left[\frac{1}{q_s+k}, \frac{1}{q_s+1}\right]$, and when q_s is large enough, $(1-\sigma)^{q_s}$ tends to e^{-1} , so the probability that C does not terminate is $\Pr[\varepsilon_1 \wedge \varepsilon_2 \wedge \varepsilon_3] \geq \frac{1}{k}\left(1-\frac{q_k}{q_C}\right)\left(1-\frac{q_{sk}}{q_C}\right)\frac{1}{e(q_s+k)}$.

In summary, if the challenger C simulation process does not terminate, and the adversary A_1 breaks the scheme of this paper with a non-negligible probability ε , the discrete logarithm problem can be successfully solved with probability $Adv(C) \geq \left(1-\frac{q_k}{q_C}\right)\left(1-\frac{q_{sk}}{q_C}\right)\frac{\varepsilon}{ke(q_s+k)}$ in the polynomial time.

Lemma 2: Under the random prediction model, if there is an A_{II} -type adversary A_2 , it can perform at most q_s signature inquiry, q_k partial partial key generation inquiry and q_{sk} private key generation inquiry, which can be ignored in time t . Probability ε breaks through the scheme of this paper, then there is such an algorithm C , which can use the algorithm to solve the probability $Adv(C) \geq \left(1-\frac{q_k}{q_C}\right)\left(1-\frac{q_{sk}}{q_C}\right)\frac{\varepsilon}{ke(q_s+k)}$ in polynomial time (where q_C is the total number of queries, n is the number of users of the aggregated signature, e is the natural log base) Discrete logarithm problem.

The proof process for the A_{II} type of the adversary A_2 is similar to the A_{I} type of the adversary A_1 . In view of the limited space, the proof process is abbreviated.

Theorem 1 Under the random prediction model, because the discrete logarithm problem is difficult, the scheme of this paper is unforgeable under the adaptive selection message attack.

V. CONCLUSION

This paper presents an efficient short signature length aggregate signature scheme, which solves the privacy protection and performance issues of the blockchain. Under this signature scheme, the length of the aggregate signature is independent of the number of users, which is fixed and reduces the storage overhead. In addition, the signature scheme constructs a signature based on the discrete logarithm problem, instead of constructing a bilinear map based, which reduces the computational overhead. At the same time, in the blockchain transaction, the identity privacy of the receiver is effectively protected.When a transaction contains n input addresses and m output addresses, the number of signatures can be reduced from n to 1.

In view of the limited space, this paper does not analyze the efficiency of the signature scheme. In the next stage, the calculation efficiency, communication efficiency and storage efficiency will be compared with the related schemes.

ACKNOWLEDGMENT

This work is supported by National Natural Science Fund for Innovative Research Groups (No.61521003), and National Natural Science Foundation of China (No.61801515).

REFERENCES

[1] Narayanan A, Bonneau J, Felten E, et al. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction[M]. Princeton University Press, 2016.

[2] Ron D, Shamir A. Quantitative Analysis of the Full Bitcoin Transaction Graph[C]// International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2013.

[3] Nakamoto S. Bitcoin: A peer - to - peer electronic cash system[J]. Consulted, 2008.

[4] Meiklejohn S, Pomarole M, Jordan G, et al. A fistful of bitcoins: characterizing payments among men with no names[C]// Proceedings of the 2013 conference on Internet measurement conference. ACM, 2013.

[5] Bonneau J, Narayanan A, Miller A, et al. Mixcoin: Anonymity for Bitcoin with Accountable Mixes[J]. 2014.

[6] Van Saberhagen N. Cryptonote v2.0[J]. 2013: 1-13.

[7] Miers I, Garman C, Green M, et al. Zerocoin: Anonymous distributed e-cash from bitcoin[J]. IEEE Symposium on Security & Privacy, 2013:397-411.

[8] Maxwell G. Coinjoin: Bitcoin privacy for the real world[J]. 2013.

[9] Liu Y, Liu X, Tang C, et al. Unlinkable Coin Mixing Scheme For Transaction Privacy Enhancement of Bitcoin[J]. IEEE Access, 2018:1-1.

[10] Wijaya D A, Liu J K, Steinfeld R, et al. Anonymizing Bitcoin Transaction[C]// International Conference on Information Security Practice and Experience. Springer International Publishing, 2016.

[11] Heilman E, Baldimtsi F, Goldberg S. Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions[C]// International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2016.

[12] Miers I, Garman C, Green M, et al. Zerocoin: Anonymous Distributed E-Cash from Bitcoin[C]// 2013 IEEE Symposium on Security and Privacy. IEEE, 2013.

[13] Sasson E B, Chiesa A, Garman C, et al. Zerocash: Decentralized Anonymous Payments from Bitcoin[C]// 2014 IEEE Symposium on Security and Privacy. IEEE, 2014.

[14] Ruffing T, Moreno-Sanchez P. ValueShuffle: Mixing Confidential Transactions for Comprehensive Transaction Privacy in Bitcoin[J]. 2017.

[15] Yuan Chao. Research on Key Technologies of Blockchain Privacy Protection [D]. Strategic Support Force Information Engineering University, 2018.

[16] Yang Tao, Kong Lingbo, Hu Jianbin, et al. Review of Polymeric Signatures and Their Applications[J]. Journal of Computer Research and Development, 2012(s2):192-199.

[17] Molloy M, Reed B. A critical point for random graphs with a given degree sequence[J]. 1995, 6(2-3):161-180.

[18] Seshadhri C, Kolda T G, Pinar A. Community structure and scale-free collections of Erd\"os-R\'enyi graphs[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2011, 85(5 Pt 2):056109.