Noname manuscript No.
(will be inserted by the editor)

Ideas

Donghang Lu

the date of receipt and acceptance should be inserted later

1 Introduction

Our goal is to design a transaction mixing protocol, which takes many transactions as input with the form of secret sharing and output them as a whole anonymity set. And validate nodes are not able to find out the source of any transaction, therefore providing privacy and unlinkability for these transactions.

And my protocol is a (n, k, t) protocol, there are n validate nodes who receive secret shares from multiple clients, and up to t of them may be corrupted by an advrsary, and any group of $t + 1$ or more validate nodes will reveal single secret, and our protocol will output k secrets each time so that these k secrets form an anonymity set and no one can trace their source.

2 Preliminaries

In this section, we discuss the assumptions and the system model for our protocols, giving special attention to their practicality over the Internet.

2.1 Communication Model

Our protocol is based on asynchronous setting. We do not depend on any time bounds and limits.

Formally, we consider an asynchronous network of $n \geq 3t+1$ parties $P_1, \ldots, P_n$ of which the adversary may corrupt up to t parties during its existence . Every pair of parties is connected by an authenticated and private communication link. Besides, there will be a number of clients $C = \{C_1, C_2, \ldots, C_t\}$ and we have no restriction about the number of clients in every protocol instance.

Address(es) of author(s) should be given

2.2 Adversary Setting and Cryptographic Assumptions

Our protocol is prove to be secure under computational security setting . Adversary $\boldsymbol{A}$ is computationally bounded with a security parameter κ. We can consider $\boldsymbol{A}$ as a probabilistic polynomial time algorithm with a security parameter k unless stated otherwise.

The adversary $\boldsymbol{A}$ is $t-bounded$ which means $\boldsymbol{A}$ can only control up to t parties in every protocol instance. A party is *honest* if it is not corrupted by adversary $\boldsymbol{A}$.

Besides, *network* can also be corrupted by adversary $\boldsymbol{A}$ and $\boldsymbol{A}$ may delay messages between any two parties. But as said before, Every pair of parties is connected by an authenticated and private communication link so $\boldsymbol{A}$ is not capable of reading or modifying messages. $\boldsymbol{A}$ cannot delay the message infinitely, which means messages have to be delivered finally by honest parties.

And since we will use $PolyCommit_{ped}$ in eAVSS-SC,which is a version of Verifiable Secret Sharing(VSS) some cryptographic assumptions are needed as below:

DLog Assumption:

Definition 1 Given a generator g of a group with prime order p and $a \in_R \mathbb{Z}_p$, for every adversary $\boldsymbol{A}_{DLog}$, $Pr[\boldsymbol{A}_{DLog}(g, g^a) = a] = \epsilon(k)$.

inserted later
t-polyDH Assumption: inserted later
t-SDH Assumption: inserted later

2.3 Verifiable Secret Sharing(VSS)

Verifiable Secret Sharing(VSS) Scheme is an important tool in our protocol. VSS avoids any party from knowing any information about transactions before they are

mixed, therefore providing privacy for these transactions.

Definition 2 An (n,t)-VSS scheme among n parties in $P = \{P_1, P_2, \ldots, P_n\}$ with a distinguished party $P_d \in P$ consists of two phases: the sharing phase(**Sh**) and the reconstruction phase (**Rec**).

Sh phase. A dealer P_d distributes a secret $s \in \mathbb{F}_p$ among parities in P. At the end of the **Sh** phase, each honest party P_i holds a share s_i of the distributed secret s.

Rec phase. In this phase, each party P_i sends its secret share s'_i to every party in P and a reconstruction function is applied in order to compute the secret $s = \mathbf{Rec}(s'_1, s'_2, \ldots, s'_n)$ or output $\perp$ indicating P_d is malicious. For honest parties $s'_i = s_i$, while for malicious parties s'_i may be different from s_i or even absent.

And a VSSscheme should have the following properties:

Secrecy: If the dealer is honest, the adversary who can compromise t parties does not have any more information about s except what is implied by the public parameters.

Correctness: If P_d is honest, the reconstructed value should be equal to the dealer's secret s.

Commitment: Even if P_d is dishonest, there exists a value s* $\in \mathbb{F}_p \cup \{\perp\}$ at the end of **Sh** phase, such that all honest parties output s^* at the end of **Rec** phase.

In this paper, we modify VSS slightly to deal with our problem. The source of the secret s will be from clients instead of parties. the responsibility of parties is to deal with these secrets and mix them, but not create them. Clients are the ones who want to share secrets to parties. This will cause slight modification of VSS protocol and will be shown later in the paper.

Also, Asynchronous VSS (AVSS)is needed since our protocol is designed for asynchronous settings. AVSS protocols require two more properties besides Secrecy, Correctness and Commitment:

Definition 3 An asynchronous VSS protocol having $n > 3t+1$ parties with a t-limited Byzantine adversary satisfies the following conditions:

Liveness: If the dealer P_d is honest in the **Sh** phase, then all honest parties complete the **Sh** phase.

Agreement: If some honest party completes the **Sh** phase, then all honest parties will eventually complete the **Sh** phase. If all honest parties subsequently start the **Rec** phase, then all honest parties will complete the **Rec** phase.

Secrecy,Correctness and Commitment: As defined before.

There are many computational VSS protocols designed for asynchronous setting: AVSS,APSS, and MPSS. Among these protocols AVSS is the first and the most pracical asynchronous VSS scheme, but its communication complexity is $O(kn^3)$. So in our protocol we choose eAVSS-SC as our sub-protocol. eAVSS-SC is an efficient VSS scheme and its communication complexity is $O(kn^2)$ by using a novel way to achieve commitment.

2.4 VBA

In our protocol we need to deal with *Byzantine agreement* problem since parties need to agree on a final anonymity set in asynchronous setting. It is impossible to solve *Byzantine agreement* by deterministic protocols so randomized protocols are necessary. The first polinomial-time solution to this problem is given by Canetti and Rabin, it can guarantee a particular outcome only if all honest parties propose the same value. *Validated Byzantine agreement*(VBA) uses *external* validity condition which is based on a global, polynomial-time computable predicate Q_{ID} known to all parties. Parties will propose value together with valiadation information. VBA ensures that the decision value satisfies Q_{ID} , and it has been proposed by at least one party.

When a server P_i starts a validated Byzantine agreement (VBA) protocol with a tag ID and input $v \in \{0,1\}*$, we say P_i proposes v for ID. When a server terminates a validated Byzantine agreement protocol with tag ID and outputs a value v, we say P_i decides v for ID.

Definition 4 (Validated Byzantine Agreement)A protocol for validated Byzantine agreement with predicate Q_{ID} satisfies the following conditions for every t-limited adversary, except with negligible probability:

External Validity: Every honest server that terminates decides v for ID such that$Q_{ID}(v)$ holds.

Agreement: If some honest server decides v for ID, then any honest server that terminates decides v for ID.

Liveness: If all honest servers have been activated on ID and all associated messages have been delivered,then all honest servers have decided for ID.

Integrity: If all servers follow the protocol, and if some server decides v for ID, then some server proposed v for ID.

Efficiency: For every ID, the communication complexity of instance ID is probabilistically uniformly bounded.

The VBA protol used in our protocol has message complexity $O(n^2)$ and communication complexity $O(n^3 + n^2(K + |v|))$ where v is the longest value proposed by any server and K is the length of a threshold signature.

3 m-VSS Protocol

3.1 construction

The protocol is devided into **Sh** phase and **Rec** phase.

From a high level point of view, **Sh** phase works in three stages. First, every client sends their secret share s_i of s to every party P_i (if client is honest) and for each secret share request with id SID , each party will start a eAVSS-SC protocol dealing with it. Second, a party P_j waits for k sharing instance $SID|eAVSS-SC$ reaching $READY$ phase, which means their ζ is consistent and at least one node P_j has already received $(echo, \zeta)$ from at least $(n-t)$ nodes for each of these k eAVSS-SC instance.Third,after the nodes decide in the VBA protocol for a set $\mathcal{L}$ which includes k eAVSS-SC instances, every nodes waits for these sharings to complete,the final share of P_j will be the summation of all these k shares.

More precisely, for each server P_i, **Sh** will following these steps:

1. for each secret share request (ID_d, $SEND$, ζ,C, $H_c(x)$, $\mathbf{W_i}$, $\phi_i(x)$, $\hat{\phi}_i(x)$) from any client, P_i initialize a eAVSS-SC instance $ID_d|eAVSS-SC$ to deal with it.

2.P_i waits for k eAVSS-SC instances entering $READY$ phase, such that in these k instances are consistent. Then P_i proposes the set of k sharing which reach $READY$ phase for validated Byzantine agreement. The proposal is a set $\mathcal{L} = \{(i, M_i)\}$, indicating every ready sharing and containing the list M_i of signatures on $echo$ messages. The predicate of VBA needs to verify that a proposal contains k valid lists of signatures from instances of protocol eAVSS-SC.

Of course if P_i is not the launcher of VBA protocol, it just participates in VBA protocol and agree on final set.

3.after the nodes decide in the VBA protocol for a set $\mathcal{L}$ which includes k eAVSS-SC instances, P_i wait for these k instances to complete. Then P_i will compute its final share: $s = \sum s_i, i \in \{1, 2, \ldots, k\}$.

Rec phase of our protocol is exactly the same as eAVSS-SC . And the result will be the summation of these k secrets $s_1 + s_2 + s_3 + ... + s_k$.

And we can simply get $s_1^2 + s_2^2 + s_3^2 + ... + s_k^2$ by doubling message size so that each message will actually contain two VSS instances, one for s_i and one for s_i^2. With the same logic we can get $s_1^3 + s_2^3 + ... + s_k^3$ until $s_1^k + s_2^k + ... + s_k^k$. With these values we can reveal all the secrets. And the result of this is the increase of communication complexity .

Protocol Sh for server P_i

upon initialization:
$\mathcal{L} = \{\}$
$cnt = 0$

upon
for each (ID_d, $SEND$, ζ, C, $H_c(x)$, $\mathbf{W_i}$, $\phi_i(x)$, $\hat{\phi}_i(x)$) from clients **do** initialize a share $ID_d|eAVSS-SC$ using protocol eAVSS-SC.

upon (ID_d, $echo$, ζ') from at least $(n-t)$ parties:
do as eAVSS-SC does
$\mathcal{L} = \mathcal{L} \bigcup \{ID_d\}$
$cnt++$
if $cnt = k$:
propose $\mathcal{L}$ in a multi-valued VBA protocol with predicate **verify signature**.
wait for VBA protocol to decide a final set $\mathcal{L}'$

upon agree on final set $\mathcal{L}'$ in VBA protocol
wait for all sharings $ID_d|eAVSS-SC$ in $\mathcal{L}'$ to finish their **Sh** phase.
calculate final secret $s = \sum s_i, i \in \{1, 2, \ldots, k\}$ where s_i is the i-th secret share in $\mathcal{L}'$.
complete **Sh** phase with (s,$\mathcal{L}'$) together with validating information as output.

Fig. 1 DKG Protocol (Pessimistic Phase)

Proof of knowledge is needed here to prove the k VSS instances in each message are corresponding to s_i,s_i^2, ... and s_i^k.

3.2 Complexity Analysis

There are two sub-protocols running in our protocol.

The message complexity and communication complexity of eAVSS-SC is $O(n^2)$ and $O(sn^2)$ where s is secure parameter. and there are k eAVSS-SC instances running in our protocol and in each eAVSS-SC instance there are actually k secrets to be shared. So the message complexity and communication complexity due to VSS is $O(kn^2)$ and $O(sk^2n^2)$. And the message complexity and communication complexity of VBA is $O(n^2)$ and $O(n^3)$ for now. So the message complexity of our m-VSS protocol is $O(kn^2)$ and $max\{O(sk^2n^2), O(n^3)\}$.

4 Analysis

The main theorem for our m-VSS is as follows.

Theorem 1 *Protocol m-VSS satisfies the following conditions for any t-limited adversary:*
Liveness: If the adversary initializes all honest servers and delivers all associated messages, and the clients

are honest throughout the sharing stage, then all honest parties complete the sharing, except with negligible probability.

Agreement: Provided the adversary initializes all honest servers and delivers all associated messages, the following holds: If some honest server completes the sharing, then all honest servers complete the sharing and if all honest servers subsequently start the reconstruction phase, then every honest server P_i reconstructs some z_i, except with negligible probability.

Privacy: If an honest client has shared s using ID_d and less than t honest servers have started the reconstruction for ID_d, the adversary has no information about s.

Sender Anonymity: : All parties including servers and adversary have no information about the source of any secret in final anonymity set. All they know is the secret is from one of the k clients with the same probability.

And the following are proofs:

Liveness: If every client P_d is honest, then every honest party will receive **send** messages sent by P_d and will start their eAVSS-SC sub-protocols. Liveness of eAVSS-SC can make sure that all these eAVSS-SC will complete **Sh** phase. So when k of these instances finish **Echo** phase, **VBA** protocol will be triggered. Liveness of **VBA** protocol indicates that all party will decide on annymity set $\mathcal{L}$. As a result eAVSS-SC instance in $\mathcal{L}$ will complete Sh Phase and all honest parties will calculate final seret share and complete **Sh** phase.

Agreement: An honest party completes **Sh** phase suggests their is a anonymity set $\mathcal{L}$ it agrees on. And Agreement of **VBA** protocol shows that if some honest party decides $\mathcal{L}$, then any honest party that terminates decides $\mathcal{L}$. So every honest party will agree on $\mathcal{L}$ and Agreement of eAVSS-SC makes sure that the k eAVSS-SC instances will be completed for every honest party. So each honest party will get k sub-shares and sum them up to get final secret share, completing $\mathcal{S}\langle$ phase.

For agreement in the **Rec** phase, all honest parties will hold an evaluation of a degree-t polynomial $\phi(x) = \sum_{i=1}^{n} \phi_i(x)$ after **Sh** phase. So in **Rec** phase if at least $2t+1$ parties join **Rec** phase, there will be at least $t+1$ honest parties with correct evaluations. With these $t+1$ evaluations the secret $s = \phi(0)$ will be reconstructed successfully.

Secrecy:

Here we will show that with honest dealers P_d the adversary $\boldsymbol{A}$ has no information about any of the secrets s. This is because of the hiding property for polynomial commitments used in eAVSS-SC. It is impossible to reconstruct polynomial $\phi(x)$ given only t messages of the form $(ID_d, SEND\ , \zeta, C, H_c(x), \mathbf{W_j}, \phi_j(x), \hat{\phi_j}(x))$. So adversary will have no infomation about $\phi(0)$ which is dealer's secret.

Sender Anonymity: insert later.

5 referance

[1] Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems