Finding Compact Reliable Broadcast in Unknown Fixed-Identity Networks (Short Paper)

Huafei Zhu and Jianying Zhou

Institute for Infocomm Research, A-star, Singapore
{huafei, jyzhou}@i2r.a-star.edu.sg

Abstract. At PODC'05, Subramanian, Katz, Roth, Shenker and Stoica (SKRSS) introduced and formulated a new theoretical problem called reliable broadcast problems in unknown fixed-identity networks [3] and further proposed a feasible result to this problem. Since the size of signatures of a message traversing a path grows linearly with the number of hops in their implementations, this leaves an interesting research problem (an open problem advertised by Subramanian et al in [3]) − how to reduce the communication complexity of their reliable broadcast protocol?

In this paper, we provide a novel implementation of reliable broadcast problems in unknown fixed-identity networks with lower communication complexity. The idea behind of our improvement is that we first transfer the notion of path-vector signatures to that of sequential aggregate path-vector signatures and show that the notion of sequential aggregate path-vector is a special case of the notion of sequential aggregate signatures. As a result, the currently known results regarding sequential aggregate signatures can be used to solve the open problem. We then describe the work of [3] in light of sequential aggregate signatures working over independent RSA, and show that if the size of an node $v_{i,j}$'s public key $|g(v_{i,j})|$ is $t_{i,j}$ and the number of hops in a path p_i is d_i in the unknown fixed-identity graph G (with k adversaries), the reduced communication complexity is approximate to $\sum_{j=1}^{d_i-1} t_{i,j}$ while the computation (time) complexity of our protocol is the same as that presented in [3].

Keywords: aggregate path-vector signatures, path-vector signatures, reliable broadcast problem.

1 Introduction

At PODC'05, Subramanian, Katz, Roth, Shenker and Stoica (SKRSS) introduced and formulated a new theoretical problem called reliable broadcast problems in unknown fixed-identity networks [3] and then proposed a feasible result such that given a bound k on the number of adversaries, there exists a distributed algorithm that achieves reliable broadcast in an unknown fixed-identity network if and only if G is $(2k+1)$ vertex connected. The key idea behind their construction is that − if a node u propagates a path-vector message (m, s, p) to

v, then v's identity is already appended to the path p by u signifying that u has propagated the message to v. As a result, the signatures' size of the message over the path grows linearly with the number of hops and thus increases communication overheads. This leaves an interesting research problem (an open problem advertised by Subramanian et al in [3]) − how to reduce the communication complexity of the proposed reliable broadcast protocol?

1.1 More on SKRSS' Assumptions

To remove the public key infrastructure (PKI) assumption, Subramanian, Katz, Roth, Shenker and Stoica further made the followings two assumptions on mobile nodes:

- **Assumption 1:** the identity of a node is fixed which cannot be forged;
- **Assumption 2:** a node knows the identities of its neighbors in the underlying graph.

We stress that the second assumption can be absorbed by a broadcasting protocol itself and thus can be implemented without any difficulty. For example, an initiator node broadcasts a message "hello, every neighbor node sends me a reply message please". Each node within one hop that receives this request for the first time appends its identifier to the received message and then sends it back to the initiator.

We also stress that an implementation of the first assumption however is a difficult task in the non-PKI setting since if the identity of a node cannot be forged, then some kinds of digital signatures or commitment schemes should be involved just like the certificates in the PKI setting. A simple way here is that we allow a device manufactory (serving as a trusted third party) to sign individual device id. The concatenation of a device id and its signature in turn is viewed as a fixed device id which can be publicly verified. Since a verifier cannot check the validation of the manufactory's public key on time in the non-PKI setting (yet this verification can be made off-line or with the help of proxies of the device, i.e., via delegation technique), we refer to this kind of signatures as obliviously committed signatures. Intuitively, an obliviously committed signature is a digital signature that is used for signing an id of a device by a third party (need not to be trusted), and the validity of the signer's public key is questionable from the point view of a verifier at the current time, and thus it is oblivious (1-out-of 2, validity (1) or invalidity (0)), however the verifier will obtain a correct answer (0 or 1) with the help of others. Since there is no satisfactory solution to this problem, we thus leave an interesting problem to the research community − how to implement secure yet efficient (with low computation and communication complexities) obliviously committed signatures in the non-PKI environment?

1.2 This Work

The goal of this paper is to provide a solution to Subramanian et al's open problem. We employ sequential aggregate signatures to accomplish this task. The contribution of this paper is twofold.

– In the first fold, we transfer the notion of path-vector signatures to that of sequential aggregate path-vector signatures and show that the notion of sequential aggregate path-vector is a special case of the notion of sequential aggregate signatures. As a result, the currently known results regarding sequential aggregate signatures can be used to solve the open problem.
– In the second fold, we describe the work of [3] in light of the best result regarding sequential aggregate signatures that are working over independent RSA moduli presented in [4] and show that assuming that $v_{i,j}$'s public key size is $|g(v_{i,j})| = t_{i,j}$ and the number of hops in a path p_i is d_i in the unknown fixed-identity graph G (with k adversaries), the reduced communication complexity is approximate to $\sum_{j=1}^{d_i-1} t_{i,j}$ while the computation (time) complexity of our protocol is the same as that presented in [3]. We thus, provide a solution to the open problem presented in [3].

2 From Path-Vector Signatures to Sequential Aggregate Signatures

2.1 Path-Vector Signatures

The basic tool used to solve the reliable broadcast problem in unknown fixed-identity networks is the notion of path-vector signatures introduced in [3]. Informally, a path-vector signature consists of the following three algorithms [2]:

– Public-key initialization: By m, we denote a message sent by v_1 to v_n over a path $(v_1, \cdots, v_n)$. Every node v_i generates its public key $g(v_i)$, and communicates it to its neighbor v_{i-1}.
– Message initialization: The source node v_1 sends the message $m_1 = [(m, s, p)$, $sig_1]$ to its neighbor v_2 where $s = (v_1, g(v_1))$, $p_1 = [(v_1, g(v_1)), (v_2, g(v_2))]$, and $sig_1 = sig((m, s, p_1), g(v_1))$;
– Incremental update: Node v_i receives message $m_{i-1} = [(m, s, p_{i-1}), sig_{i-1}]$ from its predecessor v_{i-1}. It then sends message $m_i = [(m, s, p_i), sig_i]$ to its successor v_{i+1} where $p_i = [(p_{i-1}, (v_{i+1}, g(v_{i+1})]$ and $sig_i = (sig_{i-1}$, $sig((m, s, p_i), (v_i, g(v_i))$.

We stress that the notion of a path-vector signatures presented in [3] is order specified. Furthermore, it allows multi-signer to sign same message m. As a result, the notion of a path-vector signatures is equivalent to that of order-specified multi-signature schemes and thus notion of sequential aggregate path-vector is a special case of the notion of sequential aggregate signatures.

To reduce communication complexity of the reliable broadcasting problem, we need to compress the size of the underlying path-vector signature. Motivated by this consideration, a new notion which we called sequential aggregate path-vector signatures can be introduced and formalized (where the same message m is signed by all nodes along the path p) in a natural way. We however derive this new notion from the standard notion of sequential aggregate signatures so that currently known results regarding sequential aggregate signatures can be used for our purpose, rather than formalize the stand-alone notion of sequential aggregate path-vector signatures.

2.2 Syntax of Sequential Aggregate Signatures in the Public-Key Infrastructure (PKI) Environment

Syntax. A sequential aggregate signature scheme (KG, $\mathsf{AggSign}$, AggVf) consists of the following algorithms [2]:

- Key generation algorithm (KG): On input l and k_i, KG outputs system parameters param (including an initial value $\mathcal{IV}$, without loss of generality, we assume that $\mathcal{IV}$ is a zero strings with length l-bit), on input param and user index $i \in \mathcal{I}$ and k_i, it outputs a public key and secret key pair (PK_i, SK_i) of a trapdoor one-way permutation f_i for a user i.
- Aggregate signing algorithm ($\mathsf{AggSign}$): Given a message m_i to sign, and a sequential aggregate σ_{i-1} on messages $\{m_1, \cdots, m_{i-1}\}$ under respective public keys PK_1, $\cdots$, PK_{i-1}, where m_1 is the inmost message. All of m_1, $\cdots$, m_{i-1} and PK_1, $\cdots$, PK_{i-1} must be provided as inputs. $\mathsf{AggSign}$ first verifies that σ_{i-1} is a valid aggregate for messages $\{m_1, \cdots, m_{i-1}\}$ using the verification algorithm defined below (if i=1, the aggregate σ_0 is taken to be zero strings 0^l). If not, it outputs $\perp$, otherwise, it then adds a signature on m_i under SK_i to the aggregate and outputs a sequential aggregate σ_i on all i messages $m_1, \cdots, m_i$.
- Aggregate verifying algorithm (AggVf): Given a sequential aggregate signature σ_i on the messages $\{m_1, \cdots, m_i\}$ under the respective public keys $\{PK_1, \cdots, PK_i\}$. If any key appears twice, if any element PK_i does not describe a permutation or if the size of the messages is different from the size of the respective public keys reject. Otherwise, for $j = i, \cdots, 1$, set $\sigma_{j-1} = f_j(PK_1, \cdots, PK_j, \sigma_j)$. The verification of σ_{i-1} is processed recursively. The base case for recursion is $i = 0$, in which case simply check that σ_0. Accepts if σ_0 equals the zero strings.

A sequential aggregate signature is called a sequential aggregate path-vector signature if all nodes over a path p=$(PK_1, \cdots, PK_n)$ sign any same message (i.e., $m_1 = \cdots = m_n$).

The Definition of Security. The following security definition of sequential aggregative signature schemes is due to [2]. The aggregate forger $\mathcal{A}$ is provided with a initial value $\mathcal{IV}$, a set of public keys PK_1, $\cdots$, PK_{i-1} and PK, generated at random. The adversary also is provided with SK_1, $\cdots$, SK_{i-1}; PK is called target public key. $\mathcal{A}$ requests sequential aggregate signatures with PK on messages of his choice. For each query, he supplies a sequential aggregate signature σ_{i-1} on some messages m_1, $\cdots$, m_{i-1} under the distinct public keys PK_1, $\cdots$, PK_{i-1}, and an additional message m_i to be signed by the signing oracle under public key PK. Finally, $\mathcal{A}$ outputs a valid signature σ_i of a message m_i which is associated with the aggregate σ_{i-1}. The forger wins if $\mathcal{A}$ did not request $(m_i$,

σ_{i-1}) in the previous signing oracle queries. By $\mathsf{AdvAggSign}_{\mathcal{A}}$, we denote the probability of success of an adversary.

We say a sequential aggregate signature scheme is secure against adaptive chosen-message attack if for every polynomial time Turing machine $\mathcal{A}$, the probability $\mathsf{AdvAggSign}_{\mathcal{A}}$ that it wins the game is at most a negligible amount, where the probability is taken over coin tosses of KG and $\mathsf{AggSign}$ and $\mathcal{A}$. The security of sequential aggregate path-vector signatures can be defined in a similar way as that of sequential aggregate signatures with the restriction $m_i = m$ for the j^{th} oracle query over a path $p = (PK_1, \cdots, PK_n)$.

Sequential Aggregate Signatures from RSA. In [2], Lysyanskaya, Micali, Reyzin, Shacham proposed two approaches to instantiate their generic construction from RSA trapdoor one-way permutations:

- the first approach is to require the user's moduli to be arranged in increasing order: $N_1 < N_2 < \cdots < N_t$. At the verification, it is important to check that the i-th signature σ_i is actually less than N_i to ensure the signatures are unique if H is fixed. As long as $\log(N_1) - \log(N_t)$ is constant, the range of H is a subset of Z_{N_1} whose size is the constant fraction of N_1, the scheme will be secure;
- the second approach does not require the moduli to be arranged in increasing order, however they are required to be of the same length. The signature will expanded by n bits $b_1, \cdots, b_n$, where n is the total number of users. Namely, during signing, if $\sigma_i \geq N_{i+1}$, let $b_i = 1$; else, let $b_i = 0$. During the verification, if $b_i = 1$, add N_{i+1} to σ_i before proceeding with the verification of σ_i. Always, check that σ_i is in the correct range $0 \leq \sigma_i \leq N_i$ to ensure the uniqueness of signatures.

For applications of aggregate path-vector signature schemes in reliable communication where a graph G is unknown, the choice of a claimed public key of a node v_i is completely independent on the choice of a claimed public key of another node v_j in the Internet. Thus, for any RSA-based aggregate path-vector signature that works over an unknown fixed-identity graph, a reasonable assumption should be that the sizes of all moduli are bounded by a fixed size (this requirement does not violate the unknown of underlying graph G). We stress that there is efficient implementation of sequential aggregate signatures presented in [4], which is sketched below:

Sequential aggregate signatures working over independent RSA moduli

Let H: $\{0,1\}^* \rightarrow \{0,1\}^l$ be a cryptographic hash function and $\mathcal{IV}$ be the initial vector that should be pre-described by an aggregate path-vector signature scheme. The initial value could be a random l-bit string or an empty string. Without loss of generality, we assume that the initial value $\mathcal{IV}$ is 0^l. Our aggregate path-vector signature is described as follows:

- Key generation: Each user i generates an RSA public key (N_i, e_i) and secret key (N_i, d_i), ensuring that $|N_i| = k_i$ and that $e_i > N_i$ is a prime. Let G_i:

$\{0,1\}^{t_i} \rightarrow \{0,1\}^{k_i}$, be cryptographic hash function specified by each user i, $t_i = l - k_i$.

- AggPVSig: User i is given an aggregate path-vector signature g_{i-1} and $(b_1, \cdots, b_{i-1})$, a sequence of messages $m_1, \cdots, m_{i-1}$, and the corresponding keys $(N_1, e_1), \cdots, (N_{i-1}, e_{i-1})$. User i first verifies σ_{i-1}, using the verification procedure below, where $\sigma_0 = 0^l$. If this succeeds, user i computes $H_i = H(m_1, \cdots, m_i, (N_1, e_1), \cdots, (N_i, e_i))$ and computes $x_i = H_i \oplus g_{i-1}$. Then it separates $x_i = y_i || z_i$, where $y_i \in \{0,1\}^{k_i}$ and $z_i \in \{0,1\}^{t_i}$, $t_i = l - k_i$. Finally, it computes $g_i = f_i^{-1}(y_i \oplus G_i(z_i))||z_i$. By $\sigma_i \leftarrow (g_i, b_i)$, we denote the aggregate path-vector signature(if $y_i \oplus G_i(z_i) > N_i$, then b_i =1, if $y_i \oplus G_i(z_i) < N_i$, then $b_i = 0$; again we do not define the case $y_i \oplus G_i(z_i) = N_i$ since the probability the event happens is negligible), where $f_i^{-1}(y) = y^{d_i}$ mod N_i, the inverse of the RSA function $f_i(y) = y^{e_i}$ mod N_i defined over the domain $Z_{N_i}^*$.
- AggPVf: The aggregate path-vector verification algorithm is given as input an aggregate path-vector signature g_i, $(b_1, \cdots, b_i)$, the messages $m_1, \cdots, m_i$, the correspondent public keys $(N_1, e_1), \cdots, (N_i, e_i)$ and proceeds as follows. Check that no keys appears twice, that $e_i > N_i$ is a prime. Then it computes:
  - $H_i = H(m_1, \cdots, m_i, (N_1, e_1), \cdots, (N_i, e_i))$;
  - Separating $g_i = v_i || w_i$;
  - Recovering x_i form the trapdoor one-way permutation by computing $z_i \leftarrow w_i$, $y_i = B_i(f_i(v_i) + b_i N_i) \oplus G_i(z_i)$, and $x_i = y_i || z_i$, where $B_i(x)$ is the binary representation of $x \in \mathcal{Z}$ (with k_i bits).
  - Recovering g_{i-1} by computing $x_i \oplus H_i$. The verification of (g_{i-1}, b_{i-1}) is processed recursively. The base case for recursion is $i = 0$, in which case simply check that $\sigma_0 = 0^l$.

Lemma 1. *([4]): Let $\cup_{i \in I} f_i$ be a certificated homomorphic trapdoor permutation family, the sequential aggregate signature scheme described above is secure in the random oracle model.*

2.3 Optimal Result

Let f be an RSA trapdoor one-way permutation defined over Z_N^*, where N=PQ, P and Q are two large prime numbers and $|N|$=k. Let H: $\{0,1\}^* \rightarrow \{0,1\}^l$ be a cryptographic hash function. Let $f(x) = x^e$ mod N and $f^{-1}(x) = x^d$ mod N, where $ed \equiv 1$ mod $\phi(N)$, e is a public key, and d is the correspondent secret key. Throughout this section, we assume that $l \geq k + 1$.

On input a message $m \in \{0,1\}^*$, we obtain a string $H(m) \in \{0,1\}^l$ which in turn can be rewritten as the form $qN + r$ $(= H(m))$, where $0 \leq r < N$. Let $g_1(m) = f^{-1}(r)$ and $g_2(m) = \mathcal{P}(q)$, where $\mathcal{P}$ is a padding algorithm ($\mathcal{UP}$ is the correspondent un-padding algorithm) which is further defined below.

- Padding algorithm $\mathcal{P}$: on input a random string $q \in \{0,1\}^\tau$, $\mathcal{P}$ outputs a $(l-k)$-bit codeword $0^{l-k-\tau}||q$ of q which is denoted by $\mathcal{P}(q)$;
- Un-padding algorithm $\mathcal{UP}$: on input $(l-k)$-bit string $\mathcal{P}(q)$, $\mathcal{UP}(\mathcal{P}(q))$ outputs a τ-bit string q, i.e., $\mathcal{UP}(\mathcal{P}(q))$=$q \in \{0,1\}^\tau$.

By $(N, e, H, k, l, (\mathcal{P}, \mathcal{UP}))$, we denote the public key of a signature scheme. The secret key is (d, N). Our signature scheme working in an extended domain is defined as follows:

- **Signing algorithm:** on input a message m, the signer computes $H(m)$, and then writes $H(m)$ as the form $qN + r$ $(0 \leq r < N, |N| = k)$; Finally it computes $g_1(m) \leftarrow f^{-1}(r)$ and $g_2(m) \leftarrow \mathcal{P}(q)$, where $\mathcal{P}(q)$ is a padding of q; The signature σ of message m is $(g_1(m), g_2(m))$;
- **Verification algorithm:** given a putative signature $\sigma(m)$, a verifier computes $r \leftarrow f(g_1(m))$, and $q \leftarrow \mathcal{UP}$ $(g_2(m))$; Finally, the verifier checks $H(m) \overset{?}{=} qN + r$.

Using the same technique presented in [4], we show that assuming that the RSA function is a trapdoor one-way permutation defined over Z_N^*, our signature scheme defined above is provably secure against existential forgery under an adaptive chosen-message attack in the random oracle model. This ordinary signature can be further transferred to a sequential aggregate signature without bit-expansion. For simplicity, in the following discussion, we provide our solution to the open in light of the base signature case and leave an exercise to readers in the optimal case.

3 Reliable Broadcast in Unknown Fixed-Identity Networks

Given an undirected graph G, two vertices u and v are called connected if there exists a path from u to v; Otherwise they are called disconnected. The graph G is called connected graph if every pair of vertices in the graph is connected. A vertex cut for two vertices u and v is a set of vertices whose removal from the graph disconnects u and v. A vertex cut for the whole graph is a set of vertices whose removal renders the graph disconnected. The vertex connectivity $k(G)$ for a graph G is the size of minimum vertex cut. A graph is called k vertex connected if its vertex connectivity is k or greater.

3.1 Removal of Certificated Public Keys

We stress that a collection of claimed public keys within a path-vector signature must be certificated. Thus, either a trusted third party (a certificate authority) or a public key infrastructure is required. To remove the concept of certificated identity graph G_x from path-vector signatures, a new notion called keyed-identity graph G_x is first introduced and formalized in [3]. To do so, the fixed-identity assumption is critical. The fixed-identity assumption states the following thing: each node in an undirected graph G has a unique identity it cannot fake and it knows the identities of its neighbors in G. If this assumption is not met and an adversary uses different identities to different neighbors, then for any given integer $m > 0$, there exists an m-vertex connected network G on n nodes where each node is initially aware of the identities if only its neighbors such that, a single adversary using multiple identities is sufficient to disrupt reliable broadcast in G.

With the help of fixed-identity assumption, an algorithm determining genuine keyed-identity can be proposed [3]. Suppose the underlying graph G is $2k+1$ vertex connected with k adversaries, then, between every pair of good nodes, there exists at least $k+1$ vertex disjoint paths that traverse only good nodes (the fact that adversaries can at most prove k disjoint paths to a fake node is critical for the solvability of this problem.).

3.2 Sequential Aggregate Based Broadcast Protocols

Now we can embed sequential aggregate signatures into the SKRSS asynchronous broadcast algorithm presented in [3]. That is, given a path-vector message (m, s, p) and its signature, we define the keyed identity path $P_I(m, s, p)$ associated with (m, s, p) to consist of vertices $(v_i, g(v_i))$, where v_i is the identity of a node in p and $g(v_i)$ is the public key of v_i in the signature. We borrow the notation G_x from [3] to denote the keyed-identity graph computed by a node x with a set of neighbors $N(x)$. Every good node x performs the following set of operations.

Sequential aggregate based broadcast protocol in identity-fixed networks.

- Asynchronous node wake up: A node can either begin broadcast by itself or begin transmissions upon receipt of the first message from a neighbor.
- Initiation: G_x consists of one vertex $(x, g(x))$.
- For every $u \in N(x)$, x transmits $(m(x), x, [x, u])$ to u along with its sequential aggregate signature *sas*.
- Propagation: For every path-vector message (m, s, p) with sequential aggregate signature *sas* that x receives from $u \in N(x)$, x performs:
  - Immediate-neighbor key check: Check if public-key of u in S matches the same public-key used in previous messages. If not, reject (m, s, p); if $v \in N(x) \setminus \{u\}$ appears in p, then the public-key of v should also match the one directly advertised by v.
  - Verify S using the verification algorithm of the aggregate path-vector signature.
  - Learn one vertex at a time: Accept the message only if $P_I(m, s, p)$ contains at most one new keyed identity (at the end of the path) not present in G_x. If so, update G_x with $P_I(m, s, p)$.
  - Message suppression: If $P_I(m, s, p)$ adds no new vertices or edges to G_x, ignore the message.
  - To every $u \in N(x)$, x transmits (m, s, p') where $p' = p \cup \{u\}$ after updating the signature.
- Flow computation: If the number of identity-disjoint paths to $(v, g(v))$ in G_x is at least $k+1$, then x deems v to be a genuine identity and $g(v)$ to be its public key. By identity disjoint paths, we mean that no two paths should contain two different vertices $(v, g(v))$ and $(v, g'(v))$ which share the same identity v. The immediate-neighbor key check is necessary to ensure that if

an adversary $v \in N(x)$, then v uses only a single keyed-identity $(v, g(v))$ in all its messages propagated to x. Any other message that x receives (from other neighbors) which contains the identity v is accepted only if it contains the same public key $g(v)$.

Theorem 1. *Given a bound k on the number of adversaries, the algorithm described above achieves reliable broadcast in an unknown fixed-identity network $U(n, G, N)$ if and only if G is $2k+1$ vertex connected.*

Proof. The necessary condition can be argued as follows: the structure of graph G is unknown but each node knows the identities of its neighbors in our model while the entire graph G is known to all nodes in Dolev's model [1]. Clearly, Dolev's model is a special case of our model. It follows that the assumption that G is $(2k+1)$-vertex connected is a necessary condition for reliable communication in our model.

The sufficient condition can be argued as follows: Let G' be a subgraph of G consisting of all edges between honest nodes. since the underlying graph G is $(2k+1)$-vertex connected with k adversaries, it follows that G' is at least $(k+1)$-vertex connected. If every good node u can learn all the edges in G', then it can definitely compute $(k+1)$ identity disjoint paths to every other good node and hence, can successfully determine every other good node v. Consequently, to show the proposed routing algorithm achieving reliability, it is sufficient to show that every good node will eventually learn all edges in G'. Now we consider the presence of $k > 0$ adversaries and two good nodes u and v are separated by τ hops. By the broadcasting algorithm described above, we know that at the i^{th} hop, an individual node exchanges the new sequential aggregate signature it learnt from the $(i-1)^{th}$ with its neighbors. Recursively, every good node learns all edges in G' eventually since G' is at least $(k+1)$-vertex connected subgraph.

4 Computation and Communication Complexity

Assuming that $v_{i,j}$'s public key size is $|g(v_{i,j})| = t_{i,j}$ and the number of hops in a path $p_i = \{v_{i,1}, \cdots, v_{i,d_i}\}$ is d_i. The message flow of the original SKRSS protocol is that:

- message flow generated by $v_{i,1}$: $m_{i,1} = < m_i, (v_{i,1}, g(v_{i,1})), (v_{i,2}, g(v_{i,2})) >$, and $< sig_{i,1}(m_{i,1}) >$;
- message flow generated by $v_{i,2}$: $m_{i,2} = < m_i, (v_{i,1}, g(v_{i,1})), (v_{i,2}, g(v_{i,2})), (v_{i,3}, g(v_{i,3})) >$ and $< sig_{i,1}(m_{i,1})$ and $sig_{i,2}(m_{i,2}) >$;
- $\cdots$;
- message flow generated by v_{i,d_i-1}: $m_{i,d_i-1} = < m_i, (v_{i,1}, g(v_{i,1})), \cdots, (v_{i,d_i-1}, g(v_{i,d_i-1})) >$, and $< sig_{i,1}(m_{i,1}), \cdots, sig_{i,d_i-1}(m_{i,d_i-1}) >$.

The message flow along the path p_i of our reliable broadcast protocol are that:

- message flow generated by $v_{i,1}$: $v_{i,1}$: $m_{i,1} = <m_i, (v_{i,1}, g(v_{i,1})), (v_{i,2}, g(v_{i,2})) >$, $< sig_{i,1}(m_{i,1}) >$;

- message flow generated by $v_{i,2}$: $m_{i,2} = < m_i, (v_{i,1}, g(v_{i,1})), (v_{i,2}, g(v_{i,2})), (v_{i,3}, g(v_{i,3})) >$ and $< b_{i,1}, sig_{i,2}(m_{i,2}) >$, where $b_{i,1} \in \{0, 1\}$;
- $\cdots$
- message flow generated by v_{i,d_i-1}: $m_{i,d_i-1} = < m_i, (v_{i,1}, g(v_{i,1})), \cdots, (v_{i,d_i-1}, g(v_{i,d_i-1})) >$, and $< b_{i,1}, \cdots, b_{i,d_i-2}, sig_{i,d_i-1}(m_{d_i-1})>$, where $b_{i,i} \in \{0, 1\}$, $1 \leq i \leq d_i - 2$;

By $comp_i$, we denote the communication complexity of original scheme along the path p_i; and by $\widetilde{comp}_i$, we denote the communication complexity of our scheme along the same path p_i. Thus, we have the following estimation (typically, $d_i - 1 \ll min\{t_{i,1}, \cdots, t_{i,d_i-1}\}$): $comp_i$- $\widetilde{comp}_i = t_{i,1} + \cdots + t_{i,d_i-1} - (d_i - 1)$ (the term $(d_i - 1)$ is eliminated in case that our optimal sequential aggregate signature scheme is applied).

5 Conclusion

In this paper, we have transferred the notion of path-vector signatures to that of sequential aggregate signatures and have also shown that the notion of sequential aggregate path-vector is a special case of the notion of sequential aggregate signatures. We have described the work of [3] in light of sequential aggregate signatures to realize the same functionality of path-vector signatures. We have presented alternative solution to reliable broadcast problem with nearly optimal communication complexity and thus provided an efficient solution to the open problem addressed in the introduction section.

References

1. D.Dolev: The Byzantine Generals Strike Again. J. Algorithms 3(1): 14-30 (1982)
2. A.Lysyanskaya, S.Micali, L.Reyzin, H.Shacham: Sequential Aggregate Signatures from trapdoor one-way permutations. EUROCRYPT 2004: 74-90.
3. L.Subramanian, R.Katz, V.Roth, S.Shenker, I.Stoica: Reliable broadcast in unknown fixed-identity networks. PODC2005: 342- 351.
4. H.Zhu, F.Bao, R.H.Deng: Sequential Aggregate Signatures Working over Independent Homomorphic Trapdoor One-Way Permutation Domains. ICICS 2005: 207-219