This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Optimal Key Generation Policies for MANET Security

Hussein Al-Zubaidy[1], Ioannis Lambadaris[1], Yannis Viniotis[2], Changcheng Huang[1], Ren-Hung Hwang[3]
[1]SCE-Carleton University, 1125 Colonel By Drive, Ottawa, ON, K1S 5B6 Canada
[2]ECE-North Carolina State University, Raleigh, NC, USA; [3]NCCU, Chia-Yi, Taiwan
Email: {hussein, ioannis.lambadaris, huang}@sce.carleton.ca, candice@ncsu.edu, rhhwang@cs.ccu.edu.tw

***Abstract*—In this work, we investigate the optimal key generation problem for a threshold security scheme in mobile ad hoc networks. The nodes in these networks are assumed to have limited power and critical security states. We model this problem using a closed discrete-time queuing system with L queues (one per node) randomly connected to K servers (where K nodes need to be contacted to construct a key). In this model, each queue length represents the available security-related credits of the corresponding node. We treat this problem as a resource allocation problem where the resources to be allocated are the limited power and security credits. We introduce the class of Most Balancing Credit Conserving (MBCC) policies and provide their mathematical characterization. We prove, using dynamic coupling arguments, that MBCC policies are optimal among all key generation policies; we define optimality as maximization, in a stochastic ordering sense, of a random variable representing the number of keys generated for a given initial system state.**

I. Introduction and Motivation

Mobile ad-hoc network (MANET) is a wireless network formed by an autonomous collection of mobile users with self-configuration and self-maintenance abilities. Nodes are allowed to opt in or out of the network and to move freely within the range of other nodes in the network. Each one of these nodes may act as a source, a destination or a router. This means that a node may be asked to take part in a multi-hop route, by forwarding traffic that has no relevance to itself.

Initial MANET designs have typically assumed a friendly and cooperative environment and therefore have not focused on the security issue. However, the growing interest in the deployment of MANETs in potentially hostile environments, e.g., in military and commercial applications, made security a key aspect of MANET design [1]. MANETs have no infrastructure and no centralized network management authority. This open architecture, as well as the inherent vulnerability of wireless links, make the direct application of security architectures that were initially designed for wireline networks, such as Public Key Infrastructure (PKI), impossible [8]. The PKI-based security scheme relies on the availability of infrastructure and a globally trusted Certificate Authority (CA) that provide authentication of node identity [2]. On the other hand, identity-based encryption schemes [3] allow for the use of arbitrary public key that may be derived from known identity information; such schemes eliminate the need for a public key infrastructure or a certificate authority. Furthermore, a private key can be obtained by contacting a special node called Private Key Generator (PKG). This makes an identity-based scheme less expensive than a PKI scheme, in terms of key generation and verification. The drawback of an identity-based security scheme is its reliance on a single node (the PKG that keeps the master private key) to generate and authenticate security certificate. If this node is compromised (e.g., an intruder assumes the PKG identity) or if the node runs out of power, then MANET's security will be compromised. Threshold cryptography schemes [4],[5] can be used to mitigate this threat in MANET [6]. In such schemes, the master public key is known by all nodes in the network. However, the master private key is distributed among many nodes in the network. Each one of the key-keeper nodes has partial knowledge of the master private key and therefore is unable to construct other nodes' private keys by itself. In order to obtain a private key, a user node needs to contact any K out of the L key-keeper nodes that share the secret key. This network can tolerate compromise of up to $K-1$ key-keepers.

The threshold cryptography scheme has, therefore, an obvious advantage over a single PKG scheme. However, it introduces extra load for the contacted nodes in the network. The nodes in the network may operate under different conditions (e.g., energy and security limitations) and contacting some of them may jeopardize their security or it may deplete their batteries. Therefore, the selection of the K key-keeper nodes that will be contacted (in order to construct the private key) directly affects the performance of this security scheme. An optimal policy for key generation was investigated in [7]. The authors modeled the system as a multi-armed bandit problem and concluded that an indexed policy is optimal in that it minimizes a cost function of nodes' security state. They provided a technique to compute the indexes numerically.

In this work, we also consider wireless networks with a security architecture that relies on threshold cryptography. The objective of our work is to characterize the optimal policy for key generation in this security system and prove its optimality among all feasible policies. We define the optimal key generation policy as the policy that maximizes the number of keys generated for a given initial system state. The proof is carried out using a methodology based on stochastic dominance and coupling techniques [10], [11]. Such techniques have been used frequently in wireless networks, in order to address optimization problems. Most of these optimization problems involve packet scheduling (or routing) policy optimization. In [12], the authors proved that a scheduling policy that allocates

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

the available server to its longest connected queue (LCQ), in a discrete time system of parallel queues sharing a single randomly connected server, is optimal. An extension of the model presented in [12] was investigated in [13]. The authors proved that LCQ policy is still optimal when multiple servers (instead of just one) are available for the scheduler. They assumed that a queue has the same connectivity to all servers. Furthermore, they limited the scheduler action to a single server per queue at any given time slot. The authors in [14] investigated a generalization of the model in [13] without the connectivity and the action constraints. They characterized a set of policies, namely the most balancing (MB) policies, and proved their optimality using coupling arguments. Intuitively speaking, MB policies are scheduling policies that attempt to "balance the queue sizes" at every time slot.

In this work, we provide a novel formulation of the key generation problem as a queueing optimization problem. We show that the most balancing principle presented in [14] can be used to characterize the optimal key generation policy. In particular, we prove that the Most Balancing Credit Conserving (MBCC) policies are optimal.

The paper is organized as follows: in Section II, we provide the model description. In Section III, we define the class of MBCC policies. In Section IV, we provide a theoretical proof of the optimality of this class of policies.

II. Model Description

We assume that the private key is (partially) stored in L key-keeper nodes in the MANET. Any user node in the network that wishes to obtain the key needs to contact at least K out of the L nodes; any combination of K nodes will suffice to generate the key. We make two further assumptions about the operation of the system: (a) connectivity among user and key-keeper nodes is random, and, (b) key-keeper nodes are power-limited. Both assumptions are reasonable in MANET environments. Therefore, inquiries to the key-keeper nodes may not be possible all the time and consume power.

We model the operations of this network as a discrete-time queuing system; without loss of generality, time is slotted and a key inquiry is generated every single time slot. The key-keeper nodes are modeled as queues with finite capacity. A queue holds the "credits" that remain at the node; these credits represent (energy) resources allocated for security computations. At system startup, a node contains an initial number of credits and no further credits are generated (i.e., no exogenous arrivals to the queues).

The probing operations of a user node that wishes to generate a key are depicted in Figure 1. Since probing a key-keeper node results in credit consumption, we model probing as removal of one credit from the corresponding queue. We then model successful generation of a key by having K symmetrical "servers" remove K credits from the queues. A server can remove one credit at any given time slot, i.e., one credit is used every time a node is probed. A server can only serve connected nonempty queues. The task is completed only when all the K servers are allocated. Therefore, the security key can be assembled successfully only when there are at least K connected, nonempty queues during that time slot.

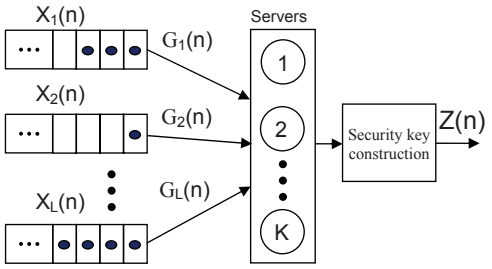

Fig. 1. Queuing model for key generation in MANET.

A. Notation

We will use **bold face**, UPPER CASE and lower case letters to represent vector quantities, random variables and sample values respectively. We define the following notation:

- $\mathbf{X}(n) = (X_1(n), X_2(n), \ldots, X_L(n))$ is the vector of queue lengths, where $X_i(n)$ represents the number of credits available in the i^{th} key-keeper node at the beginning of time slot n.
- $\mathbf{G}(n) = (G_1(n), \ldots, G_L(n))$ is the channel connectivity vector during time slot n, where $G_i(n)$ denotes the state of the channel connecting the i^{th} key-keeper node to the probing user node during the n^{th} time slot. The channel can be either connected ($G_i(n) = 1$) or not connected ($G_i(n) = 0$) (i.e., a queue is either connected to all servers or not connected at all). The connectivity process $(\{G_i(n)\}_{n=1}^{\infty}, \forall i)$ is assumed to be an i.i.d. sequence.
- $\mathbf{Q}(n) = (Q_1(n), \ldots, Q_K(n))$ is the "server allocation" vector. $Q_j(n) \in \{0, 1, \ldots, L\}$ denotes the index of the queue served by server j during time slot n. Note that we have defined a "dummy" queue, queue 0, such that $Q_j(n) = 0$ means server j is idling during time slot n.
- $\mathbf{Y}(n) = (Y_1(n), Y_2(n), \ldots, Y_L(n))$ is the "credit withdrawal" vector. $Y_i(n) \in \{0, 1\}$ is the number of credits removed from queue i during time slot n.

B. Control policies for key generation

The probing outcome at time slot n is given by:

$$Z(n) = \begin{cases} 1, & \text{if } \sum_{i=1}^{L} Y_i(n) = K, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

With this notation, a successful key generation is denoted by $Z(n) = 1$. If (due, for example, to limited connectivity) $\sum_{i=1}^{L} Y_i(n) < K$ and thus $Z(n) = 0$, a key cannot be generated. The objective of this work is to identify and study the policy that maximizes the number of successful key generations (roughly speaking, the policy that maximizes the number of times that $Z(n) = 1$). In the queuing model we proposed, this objective can be rigorously formulated as a cost function of the queue occupancies, as we shall see next.

From Equation (1), the key generation process can thus be described via the withdrawal vectors $\mathbf{Y}(n)$. Note that the variable $Y_i(n)$ is related to $Q_j(n)$ as follows:

$$Y_i(n) = \mathbb{1}_{\{i=Q_j(n)\}} \quad \forall j = 1, 2, \ldots, K \tag{2}$$

where the indicator function $\mathbb{1}_{\{\mathcal{A}\}}$ returns 1 if the statement $\mathcal{A}$ is true and 0 otherwise. Note that only one server can be allocated to a connected queue (i.e., only one credit at a time can be removed from a given key-keeper node).

Let $(\mathbf{X}(n), \mathbf{G}(n))$ denote the state of the system at the beginning of time slot n. We assume that the controller has complete knowledge of the system state information at the beginning of each time slot. We define the set $\mathcal{Y}(\mathbf{x}, \mathbf{g})$ as the set of feasible controls in state $(\mathbf{x}, \mathbf{g})$. A *feasible* control $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{X}(n), \mathbf{G}(n))$ is the one that satisfies the constraints:

$$0 \leq Y_i(n) \leq \min(X_i(n), G_i(n)) \quad \forall i = 1, \ldots, L, \tag{3}$$

$$\text{such that} \quad \sum_{i=1}^{L} Y_i(n) \leq K. \tag{4}$$

For any feasible control $(\mathbf{Y}(n))$, the system described above evolves according to the following equation

$$\mathbf{X}(n+1) = \mathbf{X}(n) - \mathbf{Y}(n), \quad n = 1, 2, \ldots \tag{5}$$

With the above notation, a key generation policy can be described as any rule that selects the sequence of withdrawal vectors $\mathbf{Y}(n)$; alternatively, due to Equation (2), we call a key generation policy a server allocation policy, since it can also be described via the sequence of server allocation vectors $\mathbf{Q}(n)$. Note that a given server allocation $\mathbf{Q}(n)$ defines the withdrawal vector $\mathbf{Y}(n)$ uniquely; however, the opposite may not be true.

III. Most-Balancing Credit-Conserving Policies

We define Π^{CCP}, the class of *Credit Conserving Policies* (CCP) as the set of policies that, at every time slot $n = 1, 2, \ldots,$ allocate the servers only when a successful outcome (i.e., $Z(n) = 1$) is ensured (i.e., when there are at least K connected non-empty queues in the system) and idle the servers otherwise.

Let $\hat{x}_i(n) = x_i(n) - y_i(n)$. Formally, a *Most Balancing, Credit Conserving* (MBCC) policy, π^{MBCC}, is defined as follows:

$$\pi^{MBCC} \in \Pi^{CCP}, \quad \text{such that}$$

$$\pi^{MBCC} = \underset{\mathbf{y}(n)\in\mathcal{Y}(\mathbf{x},\mathbf{g})}{\operatorname{argmin}} \sum_{i=0}^{L}\sum_{j=0}^{L} (\hat{x}_i(n) - \hat{x}_j(n))^2, n = 1, 2, \ldots \tag{6}$$

An MBCC policy attempts to "balance" the lengths of all queues in the system, and it does so by choosing a control $(\mathbf{y}(n))$, at every time slot, that minimizes the total square difference of queue lengths.

Notice that in this setup, we allow the policy to idle the servers when a successful outcome can not be achieved. The situations where the policy chooses to idle the servers will conserve credits when no apparent reward can be achieved.

IV. Optimality of MBCC Policies

In this section we provide a theoretical proof of the optimal allocation of security resources for private key generation in wireless ad hoc networks. We need the following definition before we present the criterion of optimality and Theorem 1, the main result.

A. Definition of Preferred Order

To prove the optimality of MBCC policies, we will need a methodology that enables comparison of the joint queue lengths as well as the number of keys generated, upto a given time, under different policies. Relations (D0)-(D4) below define the partial order $\succ_p$ (*Preferred over*) on the set $\mathcal{Z}_+^{L+1}$, where $\mathcal{Z}_+^{L+1}$ is the Cartesian product of $L+1$ instances of $\mathcal{Z}_+$, the set of non-negative integers. In the sequel, a vector $\mathbf{x} \in \mathcal{Z}_+^{L+1}$ will have the L queue lengths at some time slot n as its first L components and the total number of keys generated until n as its $(L+1)^{st}$ element, i.e., $x_{L+1} = \sum_{t=1}^{n} z(t)$.

We define the relation $\sqsupset_p$ on the set $\mathcal{Z}_+^{L+1}$ as follows: we order the first L elements of the vectors $\mathbf{x}$ and $\tilde{\mathbf{x}}$. Then $x_{[i]}$ (respectively $\tilde{x}_{[i]}$) represents the i^{th} largest component of the vector $\mathbf{x}$ (respectively $\tilde{\mathbf{x}}$), where $i \in \{1, 2, \ldots, L\}$. The last component remains the same in the new ordering (i.e., $x_{[L+1]} = x_{L+1}$ and $\tilde{x}_{[L+1]} = \tilde{x}_{L+1}$). Furthermore, we have

$$\tilde{\mathbf{x}} \sqsupset_p \mathbf{x} \Longleftrightarrow \begin{cases} \tilde{x}_{L+1} = x_{L+1} + 1, \\ x_{[i]} - 1 \leq \tilde{x}_{[i]} \leq x_{[i]}, \ \forall i = 1, \ldots, K, \\ \tilde{x}_{[i]} = x_{[i]}, \forall i = K+1, \ldots, L. \end{cases} \tag{7}$$

To offer some intuition regarding the relation $\sqsupset_p$, let $\mathbf{x}$ and $\tilde{\mathbf{x}}$ represent queue length (and keys generated) vectors, as defined earlier, in the systems S and $\tilde{S}$ respectively during $t = n$. Then $\tilde{\mathbf{x}} \sqsupset_p \mathbf{x}$ if (a) $\tilde{S}$ has one more key generated (than in S) by time slot n, (b) $\tilde{S}$ has at most K credits less than S, (c) this difference is a result of a reduction of at most one credit per queue for each (or some) of the longest K queues, and (d) the $L-K$ shortest queues are the same in both systems. We call this relation a "key generation interchange" since $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by interchanging a key for K or less credits removed from the K longest queues in the system.

We say $\tilde{\mathbf{x}} \geq_r \mathbf{x}$ if (D0) and any one of (D1) - (D4) hold:

(D0) $\tilde{x}_{L+1} \geq x_{L+1}$ (comparison of generated keys);

(D1) $\tilde{x}_i \geq x_i$, where $i \leq L$ (point-wise comparison of queue lengths);

(D2) $\tilde{\mathbf{x}}$ is a permutation of $\mathbf{x}$. The two vectors differ only in two (queueing) components i and j where $i, j \leq L$, such that $\tilde{x}_i = x_j$ and $\tilde{x}_j = x_i$;

(D3) $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a *"balancing interchange"*. The two vectors differ in two (queuing) components i and j only, where $i, j \leq L$, where $x_j \leq \min(\tilde{x}_i, \tilde{x}_j) \leq \max(\tilde{x}_i, \tilde{x}_j) \leq x_i$, such that: $\tilde{x}_i = x_i - 1$ and $\tilde{x}_j = x_j + 1$.

(D4) $\tilde{\mathbf{x}} \sqsupset_p \mathbf{x}$. $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a "key generation interchange".

The transitive closure of the relation $\geq_r$ defines the partial order $\succ_p$ over the set $\mathcal{Z}_+^{L+1}$, we call it the "preferred order";

when $\tilde{\mathbf{x}} \succ_p \mathbf{x}$ we say that "$\tilde{\mathbf{x}}$ is preferred over $\mathbf{x}$". Intuitively, the vector $\tilde{\mathbf{x}}$ corresponds to a system in which at least the same number of keys have been generated (at some time n), when compared to a system represented by the vector $\mathbf{x}$. Furthermore, $\tilde{\mathbf{x}}$ can be obtained from $\mathbf{x}$ by performing a sequence of reductions, permutations (of two components at a time), balancing interchanges and key generation interchanges, of its (i.e., vector $\mathbf{x}$'s) first L components.

1) The class of reward functions $\mathcal{F}$: Let $\tilde{\mathbf{x}}, \mathbf{x} \in \mathcal{Z}_+^{L+1}$. The class of reward functions, $\mathcal{F}$, we consider here, contains real-valued functions $f : \mathcal{Z}_+^{L+1} \longrightarrow \mathcal{R}$ that are monotone, non-decreasing with respect to the partial order $\succ_p$; i.e.,

$$f \in \mathcal{F} \;\; \textit{iff} \;\; \tilde{\mathbf{x}} \succ_p \mathbf{x} \Rightarrow f(\tilde{\mathbf{x}}) \geq f(\mathbf{x}) \tag{8}$$

An example function $f_1 \in \mathcal{F}$ is the total number of keys generated up to a given time n, i.e., $f_1(\mathbf{x}) = x_{L+1}$. Another example would be the function $f_2(\mathbf{x}) = \sum_{i=1}^{L} x_i + K x_{L+1}$ as well as time averages of both functions.

B. The Proof of MBCC Optimality

In this section we present and prove our main result, the optimality of MBCC policies. The following theorem states this optimality, with respect to cost functions $f \in \mathcal{F}$, among all feasible policies. In the following, $\mathbf{X}^{MBCC}$ and $\mathbf{X}^{\pi}$ represent the queue sizes as well as the number of keys generated (as defined in section IV-A) under an MBCC and an arbitrary policy π. For two real-valued random variables A and B, $A \leq_{st} B$ defines the usual stochastic ordering [10].

Theorem 1. *An MBCC policy dominates any arbitrary policy among the set of all feasible policies Π for the distributed security key management problem, i.e.,*

$$f(\mathbf{X}^{\pi}(t)) \leq_{st} f(\mathbf{X}^{MBCC}(t)), \quad \forall\, t = 1, 2, \ldots \tag{9}$$

and for all $\pi \in \Pi$ and all cost functions $f \in \mathcal{F}$.

Proof: From (8) and the definition of stochastic dominance, it is sufficient to show that $\mathbf{X}^{MBCC}(t) \succ_p \mathbf{X}^{\pi}(t)$ for all t and all sample paths in a suitable sample space and when the systems under both policies start from the same initial state. The sample space is the standard one used in stochastic coupling methods [11].

For any fixed $n \geq 1$, let Π_n denote the set of policies that have the MBCC property (i.e., behave similar to an MBCC policy) at time slots $t = 1, 2, \ldots, n$. We can easily see that these sets form a monotone sequence, with $\Pi_n \subseteq \Pi_{n-1}$. Then the set Π^{MBCC} can be defined as $\Pi^{MBCC} = \bigcap_{n=1}^{\infty} \Pi_n$.

Denote by $\mathbf{Y}^{\pi}(n)$ (respectively $\mathbf{Y}^{MBCC}(n)$) the withdrawal vector under policy π (respectively under an MBCC policy) during time slot n. For any integer $0 < h \leq K$, we say that "π deviates from MBCC at time n by at most h", if at most h server reallocations are required to make π have the MBCC property (i.e., have $\mathbf{Y}^{\pi}(n) = \mathbf{Y}^{MBCC}(n)$ and $Z^{\pi}(n) = Z^{MBCC}(n)$). Let Π_n^h be the set of policies that have the MBCC property before time slot n and deviate from MBCC at time n by at most h; we define $\Pi_n^0 = \Pi_n$. By definition, $\{\Pi_n^h\}_{h=0}^{K}$ forms a monotone sequence, with $\Pi_n^0 \subseteq \cdots \subseteq \Pi_n^{h-1} \subseteq \Pi_n^h \subseteq \Pi_n^{h+1} \subseteq \cdots \subseteq \Pi_n^K$. We can also easily check that $\Pi_n^K = \Pi_{n-1}$; note that the set Π of all policies can be expressed as $\Pi = \Pi_0 = \cup_{h=0}^{K} \Pi_1^h$.

To prove the optimality of an MBCC policy, π^{MBCC}, we start with an arbitrary policy π and apply a series of modifications that result in a sequence of policies $(\pi_1, \pi_2, \ldots)$. The modified policies have the following properties: (a) $\pi_i \in \Pi_i$, i.e., policy π_i has the MBCC property at time slots $t = 1, 2, \ldots, i$, (b) π_j dominates π_i for $j > i$ (and thus π_j has the MBCC property for a longer period of time than π_i), and, (c) π_1 dominates the given policy π, where policy domination is defined in Equation (9). We will need the following lemma to complete the proof of Theorem 1.

Lemma 1. *For any arbitrary policy $\pi \in \Pi_\tau^h$, a policy $\tilde{\pi} \in \Pi_\tau^{h-1}, h > 0$ can be constructed such that $\tilde{\pi}$ dominates π.*

Now, we will proceed with the proof of Theorem 1. Let π be any arbitrary policy; then $\pi \in \Pi_1^K$. We construct a sequence of policies starting from π by applying Lemma 1 repeatedly. Each of these policies dominates the previous one. According to Lemma 1, we obtain policies that belong to $\Pi_1^K, \Pi_1^{K-1}, \ldots, \Pi_1^0 = \Pi_1$. The last such policy, call it π_1, belongs to Π_2^K. If we continue in such manner, we will obtain a policy π_n that belongs to Π_n for increasing values of n. From the construction of π_n, we can see that it resembles the preceding policy $\pi_{n-1} \in \Pi_{n-1}$ until time $n-1$ in that both have the MB property until time slot $n-1$.

For any arbitrarily large value of n, this sequence of policies defines a limiting policy π^* that for every n agrees with π_n until time n. This means that π^* acts similar to π^{MBCC} at all times and dominates all the previous policies, including π. ■

C. Proof of Lemma 1

In this section we provide a proof for lemma 1. We will use a coupling argument [11] to prove this lemma.

Proof: Consider an arbitrary policy $\pi \in \Pi_\tau^h$ and an arbitrary sample path $\omega = (\mathbf{x}(1), \mathbf{g}(1), \mathbf{g}(2), \ldots)$, where $\mathbf{x}(.), \mathbf{g}(.)$ and $\mathbf{z}(.)$ are sample values of the random variables $\mathbf{X}(.), \mathbf{G}(.)$. We will refer to this system by S. For the remainder of this proof, we redefine the vector $\mathbf{x}(n), \forall n$ by adding the element $x_{L+1}(n)$ as defined earlier. Starting from π and ω, we construct another policy $\tilde{\pi} \in \Pi_\tau^{h-1}$ and a sample path $\tilde{\omega}$ (by selecting appropriate connectivity patterns and couple them with those in ω) such that:

$$\tilde{\mathbf{x}}(t) \succ_p \mathbf{x}(t) \tag{10}$$

for every sample path and at every time slot. We will do that in two steps: In step 1, we will construct $\tilde{\pi}$ and $\tilde{\omega}$ for times up to $t = \tau$. In step 2 we complete this construction for $t > \tau$. We refer to the new system as $\tilde{S}$.

Step 1: For the construction of $\tilde{\omega}$, we let the connectivity patterns be the same in both systems at all time slots before τ, i.e., $\tilde{\mathbf{g}}(t) = \mathbf{g}(t)$ for all $t < \tau$. We construct $\tilde{\pi}$ such that it chooses the same withdrawal vector as π, i.e., we set $\tilde{\mathbf{y}}(t) = \mathbf{y}(t)$ for all $t < \tau$. In this case, at $t = \tau$, the resulting queue

sizes as well as the number of keys generated (up to $\tau-1$) in both systems are equal, i.e., $\tilde{z}(\tau-1)=z(\tau-1)$, hence, $\tilde{\mathbf{x}}(\tau+1)=\mathbf{x}(\tau)$. Therefore at $t=\tau$, properties (D0) and (D1) of the preferred order hold true (with strict equality) and (10) is satisfied.

At time slot τ, one of the following two cases may apply:
Case 1) There are less than K connected, non-empty queues in the system at $t=\tau$. Let $\tilde{\omega}$ have the same connectivity pattern as ω, i.e., let $\tilde{\mathbf{g}}(\tau)=\mathbf{g}(\tau)$. In this case, a MBCC policy (π^{MBCC}) would idle all the servers to conserve credits, since key generation is impossible. Since π deviates from π^{MBCC} by at most h server allocation ($h>0$ by assumption), then π must allocate at least one server and at most h servers ($h<K$ by feasibility constraints and since there are less than K connected non-empty queues) and deplete h credits without generating a key, i.e., $z(\tau)=z(\tau-1)$. Let server k be a server that is allocated by π to a queue j during time slot $t=\tau$. Then we construct $\tilde{\pi}$ such that; $\tilde{y}_j(\tau)=y_j(\tau)-1=0$ (i.e., server j is idled by $\tilde{\pi}$) and $\tilde{y}_i(\tau)=y_i(\tau),\forall\, i\neq j$. In this case, $\tilde{x}_j(\tau+1)=x_j(\tau+1)+1$ and $\tilde{x}_i(\tau+1)=x_i(\tau+1),\forall\, i\neq j$ and $i\leq L+1$. Therefore, D0 holds with strict equality, D1 holds with strict inequality and Equation (10) follows. Furthermore, $\tilde{\pi}\in\Pi_\tau^{h-1}$ i.e., $\tilde{\pi}$ deviates from π^{MBCC} by at most $h-1$ at $t=\tau$.
Case 2) There are at least K connected non-empty queues in the system during time slot $t=\tau$. Two sub-cases must be considered in this case:

2a) The policy π does not generate a key at $t=\tau$ (i.e., $z(\tau)=z(\tau-1)$). This happens when at least one of the h servers that π allocates differently than π^{MBCC} is idled by policy π. We define the set Φ_S^τ to contain the collection of such servers and the set Φ_Q^τ to contain the queues that these servers were allocated to under π^{MBCC}, i.e., $j=q_k^*(\tau),\forall k\in\Phi_S^\tau$ and $j\in\Phi_Q^\tau$, where $\mathbf{q}^*(\tau)$ is the server allocation control under the policy π^{MBCC} at $t=\tau$. Let the set $\Phi_{Q'}^\tau$ contain the M longest queues in $\tilde{S}$ that are not allocated a server under π (in S) at $t=\tau$, where $M=|\Phi_S^\tau|$. Then

$$M=|\Phi_S^\tau|=|\Phi_Q^\tau|\leq h\leq K \tag{11}$$

The equality follows from the feasibility constraints, namely, the condition that one server may be allocated to one queue only at any given time slot. The leftmost inequality is apparent from the assumption that π differs from π^{MBCC} by at most h server allocations and the definition of the set Φ_S^τ. The rightmost inequality follows from Equation (3).

In $\tilde{S}$ we can generate a key during time slot τ by allocating the servers that were idled by π to their longest connected queues under $\tilde{\pi}$. We construct $\tilde{\omega}$ and $\tilde{\pi}$ at $t=\tau$ as follows:

- For every server $k\in\Phi_S^\tau$, and every queue pair $i\in\Phi_Q^\tau$ and $j\in\Phi_{Q'}^\tau$, let $\tilde{g}_j(\tau)=g_i(\tau)$ and $\tilde{g}_i(\tau)=g_j(\tau)$. Let $\tilde{\pi}$ allocate server k to queue j, i.e., $\tilde{q}_k(\tau)=j$.
- For every queue $m\notin\Phi_Q^\tau\cup\Phi_{Q'}^\tau$, let $\tilde{g}_m(\tau)=g_m(\tau)$. Let $\tilde{\pi}$ take the same action as π, i.e., $\tilde{y}_m(\tau)=y_m(\tau)$.

In this case, $\tilde{y}_i(\tau)=y_i(\tau),\forall i\notin\Phi_{Q'}^\tau$, while $\tilde{y}_j(\tau)=y_j(\tau)+1,\forall j\in\Phi_{Q'}^\tau$. Then $\tilde{x}_i(\tau+1)=x_i(\tau+1)$ and $\tilde{x}_j(\tau+1)=x_j(\tau+1)-1,\forall i,j\leq L$, and since a key is generated by $\tilde{\pi}$ in this case (all idling servers under π were allocated by $\tilde{\pi}$), then $\tilde{x}_{L+1}(\tau+1)=x_{L+1}(\tau+1)+1$. Therefore, D0 and D4 are satisfied and Equation (10) holds.

2b) The policy π generates a key at $t=\tau$ (i.e., $z(\tau)=z(\tau-1)+1$). This means that all the K servers were allocated by π. Let $\tilde{\omega}$ have the same connectivity pattern as ω, i.e., let $\tilde{\mathbf{g}}(\tau)=\mathbf{g}(\tau)$. We consider the following two cases:

(i) During time slot $t=\tau$, the original policy π deviates from a MBCC policy by *strictly* less than h server allocations. Then π belongs to Π_τ^{h-1} as well, so we set $\tilde{\mathbf{y}}(\tau)=\mathbf{y}(\tau)$. In this case, the resulting queue sizes $\tilde{\mathbf{x}}(\tau+1),\mathbf{x}(\tau+1)$ as well as keys generated $\tilde{z}(\tau),z(\tau)$ will be equal, properties (D0) and (D1) hold true and (10) is satisfied at $t=\tau+1$.

(ii) During time slot $t=\tau$, π deviates from a MBCC policy by *exactly* h server allocations. Since $\pi\in\Pi_\tau^h$ and $h>0$, we can identify a server k and two queues l and s such that: (a) both queues are connected to server k, (b) $q_k(\tau)=s$, i.e., π allocates server k to queue s, (c) no server is allocated to queue l during time slot τ by the policy π, i.e., $y_l(\tau)=0$, (d) $x_s(\tau+1)<x_l(\tau+1)$, i.e., the resulting queue l (after applying π at $t=\tau$) is "longer" than queue s.

We construct $\tilde{\pi}$ in this case by performing a "balancing interchange" as follows. During time slot τ, let $\tilde{\pi}$ be identical to π except that server k is allocated to queue l under $\tilde{\pi}$ instead of queue s. In other words, $\tilde{q}_k(\tau)=l$, $\tilde{q}_m(\tau)=q_m(\tau)$ for all $m\neq k$. From Equation (2) we can easily compute that the withdrawal vectors under the two policies satisfy

$$\begin{aligned}\tilde{y}_i(\tau) &= y_i(\tau) \text{ for all } i\neq l,s,\\ \tilde{y}_l(\tau) &= y_l(\tau)-1 \text{ and } \tilde{y}_s(\tau)=y_s(\tau)+1\end{aligned} \tag{12}$$

From the construction of $\tilde{\pi}$, it is clear that this policy deviates from π^{MBCC} during $t=\tau$ by at most $h-1$ server allocations. From Equations (12) and (5), we have $\tilde{x}_l(\tau+1)=x_l(\tau+1)-1$ and $\tilde{x}_s(\tau+1)=x_s(\tau+1)+1$. It is also evident that $\tilde{x}_i(\tau+1)=x_i(\tau+1),\forall i\neq l,s$. By assumption, we also have $x_l(\tau+1)>x_s(\tau+1)$. Therefore, $\mathbf{x}(\tau+1)$ and $\tilde{\mathbf{x}}(\tau+1)$ will satisfy property (D3). Property (D0) is also satisfied since both system generate a key during this time slot, thus (10) is satisfied at $t=\tau+1$.
Step 2: The above concluded the construction of $\tilde{\omega},\tilde{\pi}$ during $t=\tau$. The next step is to construct $\tilde{\omega},\tilde{\pi}$ for times $n>\tau$, such that the partial order (10) is still preserved. This ordering was already established for $t=\tau+1$. Now, assume that $\tilde{\pi}$ is defined up to time $n-1$ and such that $\tilde{\mathbf{x}}(n)\succ_p\mathbf{x}(n)$. We will prove that at time slot n, $\tilde{\pi}$ can be constructed such that $\tilde{\mathbf{x}}(n+1)\succ_p\mathbf{x}(n+1)$. Then using a forward induction argument, one can conclude that (10) holds for all times.

The preferred order $\succ_p$ holds when property (D0) and either D1 – D4 hold. The following four cases, that correspond to properties (D1), (D2), (D3) and (D4) are to be considered next; in all cases D0 is assumed to hold during time slot n.

Case (1) $\tilde{\mathbf{x}}(n)\geq\mathbf{x}(n)$, i.e., D1 holds. For the construction of $\tilde{\omega}$, we set $\tilde{\mathbf{g}}(n)=\mathbf{g}(n)$. We construct $\tilde{\pi}$ such that $\tilde{\mathbf{y}}(n)=\mathbf{y}(n)$. In this case, its obvious that $\tilde{\mathbf{x}}(n+1)\geq\mathbf{x}(n+1)$ and Equation (10) holds at $t=n+1$.

Case (2) $\tilde{\mathbf{x}}(n)$ is a permutation of $\mathbf{x}(n)$ (according to D2), such that $\tilde{\mathbf{x}}(n)$ can be obtained from $\mathbf{x}(n)$ by permuting components i and j, where $i, j \leq L$. For the construction of $\tilde{\omega}$, we set $\tilde{g}_i(n) = g_j(n)$ and $\tilde{g}_j(n) = g_i(n)$; the connectivities for each one of the remaining queues are the same as in ω. We construct $\tilde{\pi}$ such that $\tilde{y}_i(n) = y_j(n)$, $\tilde{y}_j(n) = y_i(n)$ and $\tilde{y}_r(n) = y_r(n)$ for all $r \neq i, j$. As a result, $\tilde{\mathbf{x}}(n+1)$ and $\mathbf{x}(n+1)$ satisfy properties (D0) and (D2) again and (10) is satisfied at $t = n+1$.

Case (3) $\tilde{\mathbf{x}}(n)$ is obtained from $\mathbf{x}(n)$ by performing a balancing interchange as defined in property (D3). In this case $x_i(n) \geq x_j(n) + 1$, by the definition in (D3). There are two cases to consider:

(3.a) $x_i(n) = x_j(n) + 1$. This case is possible only if $\mathbf{x}(n)$ and $\tilde{\mathbf{x}}(n)$ have property (D2) , i.e., they have components i and j permuted and all other components are the same. This case has already been considered in Case 2 above and the same construction of $\tilde{\omega}$, $\tilde{\pi}$ is still valid here.

(3.b) $x_i(n) > x_j(n)+1$. Let $\tilde{\mathbf{g}}(n) = \mathbf{g}(n)$. Then let $\tilde{\mathbf{y}}(n) = \mathbf{y}(n)$ and therefore D0 holds. In this case, D3 will also be preserved and (10) holds again at $t = n+1$.

Case (4) $\tilde{\mathbf{x}}(n)$ is obtained from $\mathbf{x}(n)$ by performing a key generation interchange as defined in property (D4), i.e., $\tilde{\mathbf{x}} \sqsupset_p \mathbf{x}$ as in (7). Let Υ be the set of the K longest queues in $\tilde{S}$ at time slot n. We define the set $\Psi \subseteq \Upsilon$ as the set that contains all queues i such that $\tilde{x}_i = x_i - 1$. The existence of this set is apparent from (7). If Ψ is empty then this case reduces to D1. Consider the following cases:

(4.a) If $|\Psi| > 0$ and $\tilde{x}_{i'}(n) = 0$, for some $i' \in \Psi$, according to the definition of Ψ, we must have $x_{i'}(n) = 1$; let $\Psi' \subseteq \Psi$ represent the set of these empty queues in $\tilde{S}$. Depending on the connectivity state and π during $t = n$ time slot, we may construct $\tilde{\pi}$ at $t = n$ as follows:

(i) If there are less than K connected, non-empty queues in S or if π idles some servers then it is not possible to generate a key under π. Then let $\tilde{\mathbf{g}}(n) = \mathbf{g}(n)$ and $\tilde{\mathbf{y}}(n) = \mathbf{y}(n)$. It can be easily verified that D4 holds again in this case, D0 also holds since no key is generated in either system and (10) is satisfied at $t = n+1$.

(ii) If there are K (or more) connected, non-empty queues in S and if π generates a key by allocating the K servers to K queues not in the set Ψ', then the same construction in (i) above is used here, thus D0 and D4 will hold again. Hence, (10) is satisfied at $t = n+1$ in this case too.

(iii) If there are K (or more) connected, non-empty queues in S and if π generates a key by allocating the K servers to K queues, some of which belong to the set Ψ', then let Ψ'' be the set of such queues, i.e., $\Psi'' \subseteq \Psi'$. Let $W = |\Psi''|$. Consider the following cases:

(iii-a) If there are at least K non-empty queues in $\tilde{S}$, then choose the $W \leq K$ longest connected queues in $\tilde{S}$ and put them in a new set call it Ψ^*. Let $\tilde{g}_{j^*}(n) = g_{i''}(n)$ and $\tilde{g}_{i''}(n) = g_{j^*}(n)$ for all $i'' \in \Psi''$ and $j^* \in \Psi^*$. Let $\tilde{g}_r(n) = g_r(n)$ for all $r \notin \Psi'' \cup \Psi^*$. Furthermore, let $\tilde{y}_{j^*}(n) = y_{i''}(n)$, $\tilde{y}_{i''}(n) = y_{j^*}(n)$ and $\tilde{y}_r(n) = y_r(n)$, $\forall r \notin \Psi'' \cup \Psi^*$. D4 is satisfied at $t = n+1$, $\tilde{\pi}$ also generates a key in this case, therefore D0 is also satisfied, and (10) follows.

(iii-b) If, on the other hand, there are less than K non-empty queues in $\tilde{S}$, then let $\tilde{\pi}$ idle all servers. This case is possible only at the stopping time, i.e., no more keys can be generated. It is apparent from the definition of D4 that at $t = n+1$, we have $\tilde{z}(n) = z(n)$ (i.e., D0 is satisfied) and $t = n+1$ is the stopping time for S too. With some reordering of the queues, we can show that D1 is satisfied at $t = n+1$ in this case and (10) follows.

(4.b) If $\tilde{x}_{i'}(n) > 0$ for all $i' \in \Psi$ then let $\tilde{\mathbf{g}}(n) = \mathbf{g}(n)$ and $\tilde{\mathbf{y}}(n) = \mathbf{y}(n)$. Then D0 and D4 hold at $t = n+1$ and (10) follows.

The above completes the construction of $\tilde{\pi}$. Note that policy $\tilde{\pi}$ belongs to Π_τ^{h-1} by construction in Step 1; its dominance over π follows easily from relation (8). ■

V. Conclusion

We studied the problem of optimal key generation for a threshold security scheme in MANET. We defined the set of most-balancing, credit-conserving (MBCC) policies and proved their optimality among all feasible key generation policies. The optimality is defined as maximization, in stochastic ordering sense, of the number of security keys generated for a given initial state. We used a stochastic dynamic coupling argument to prove the optimality of MBCC policies. This set of policies have potential applicability in many other optimal control problems in scheduling and resource management.

References

[1] J. Hubaux, L. Buttyan, S. Capkun, "The quest for security in mobile ad hoc networks," in Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), 2001.

[2] A. Boldyreva, M. Fischlin, A. Palacio, B. Warinschi, "A closer look at PKI: Security and efficiency," in: Public Key CryptographyPKC 2007. Lecture Notes in Computer Science, vol. 4450, pp. 458475. Springer, Heidelberg, 2007.

[3] A. Shamir, "Identity-based cryptosystems and signature schemes", Proc. Crypto 84, pp. 4753, 1984.

[4] R. L. Rivest, A. Shamir, and Y. Tauman, "How to Share a Secret," Communications of the ACM, 22(11):612613, 1979.

[5] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," CRYPTO89, Santa Barbara, CA, USA, Aug. 1989.

[6] H. Deng, A. Mukherjee, and D. Agrawal, "Threshold and identity-based key management and authentication for wireless ad hoc networks," in Proc. ITCC04, Washington, DC, USA, Apr. 2004.

[7] R.F. Yu and H. Tang, "Distributed Node Selection for Threshold Key Management with Intrusion Detection in Mobile Ad Hoc Networks," to appear in ACM/Springer Wireless Networks, 2010.

[8] H. Yang, H. Luo, F. Ye and L. Zhang, "Security in mobile ad-hoc networks: challenges and solutions," IEEE Wireless Communications, 11(1): 38-47, 2004.

[9] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," IEEE Networks, 13(6):2430, 1999.

[10] D. Stoyan, *Comparison Methods for Queues and other Stochastic Models*, J. Wiley and Sons, Chichester, 1983.

[11] T. Lindvall, *Lectures on the coupling method,* New York: Wiley(1992).

[12] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," IEEE Transactions on Information Theory, 39(2): 466 - 478, Mar. 1993.

[13] A. Ganti, E. Modiano and J. N. Tsitsiklis, "Optimal transmission scheduling in symmetric communication models with intermittent connectivity," IEEE Trans. on Info. Theory, 53(3): 998-1008, Mar. 2007.

[14] H. Al-Zubaidy, I. Lambadaris and I. Viniotis, "Optimal Resource Scheduling in Wireless Multi-service Systems With Random Channel Connectivity," IEEE Globecom09, Honolulu, HI, USA, Dec. 2009.