ChainFarm: A Novel Authentication Protocol for High-rate Any Source Probabilistic Broadcast

Ying Huang*, Wenbo He†, Klara Nahrstedt*
*Department of Computer Science, University of Illinois at Urbana-Champaign, Email: {huang23, klara}@cs.uiuc.edu
†Department of Computer Science, University of New Mexico, Email: wenbohe@cs.unm.edu

***Abstract*—Broadcast communication prevails for data dissemination and resource discovery. In mission-critical applications, extensive information sharing and coordination endow broadcast with new features: a large number of active broadcast sources, probabilistic broadcast reception and high receiving rate. We identify this type of broadcast traffic as $ASPBcast$ traffic. Many efforts have been made to authenticate broadcast source and prevent content modification in a light-weighted way using one-way hash chain (TESLA). However, they do not scale to a large number of senders. In addition, authentication delay increases under packet losses and probabilistic broadcast. The longer authentication is postponed, the longer packets are buffered, which poses a memory-based denial-of-service (DoS) threat. In this paper, we will present an efficient authentication protocol for $ASPBcast$ traffic, called $ChainFarm$. We propose an algorithm for parameter configuration to satisfy both memory and delay requirements with maximal resilience against compromise. Simulation results agree with our analysis and show distinct performance improvement.**

I. Introduction

Broadcast is heavily utilized in data dissemination and resource discovery, like command delivery, multiparty teleconference, and alert announcement. Extensive information sharing and coordination endow broadcast with new properties in today's mission-critical applications, which are identifies as High-rate Any Source Probabilistic Broadcast ($ASPBcast$ for short). The number of active broadcast sources is *large* in closely collaborative environment since every person and every device are willing and able to contribute collectively. They continuously broadcast to provide real-time monitoring and timely resource discovery. In order to cope with problems of broadcast storm, mobility and diverse interests, flooding is superseded by advanced protocols, such as gossip [1][2], mobility-assisted information diffusion [3], rendezvous-point store and query [4][5]. In these schemes, deterministic broadcast becomes probabilistic. Sources talk to a small subset of nodes each time. The contact set is randomly determined and varies with time. This uncertainty of contact set may result from errorness of wireless reception, link break due to movement, network partition and application choice in favor of randomness and robustness against a single point of failure or attacks. In summary, $ASPBCast$ has the following attributes: (1) any node could broadcast; (2) packets reception from a source is probabilistic; (3) packet reception rate per node is high due to integrated traffic from all sources.

There is an emerging need to secure communication in mobile wireless networks for mission-critical applications, such as military operation, emergency response and disaster recovery. Attackers may intend to modify or forge broadcast messages so as to force unnecessary or incorrect operations. Therefore, it is indispensable to authenticate broadcast traffic, verifying both source identity and content integrity. Nevertheless, providing broadcast authentication in mobile wireless networks is not a trivial task. The challenges come from the limited resources of wireless devices, including memory, bandwidth, CPU power and battery. *Public-key based authentication*, designed for high-end workstations, incurs heavy computation and memory overhead during the processes of signature generation and validation. In resource-constrained devices, the usual processing time for a public key signature is seconds or tens of milliseconds. Thereby, public-key based authentication cannot sustain high traffic rate and is vulnerable to DoS attacks on energy, CPU and memory resources. On the contrary, *one-way hash function* excels in low computational overhead. One of the most popular protocols is *TESLA* [6]. TESLA and its variations have been widely used in context of data streaming and sensor networks [7][8]. Authentication keys are organized in one-way hash chains. Delayed key disclosure closely mimics the authentication semantics of public-key cryptography. One distinctive hash chain is associated with a single identity to authenticate both source identity and content integrity. TESLA is featured with low communication and computation overhead, scalability to a large number of receivers and tolerance to packet loss. However, TESLA and its variations are not flexible in coordinated environment with probabilistic broadcast and high-mobility, because (1) despite its scalability to a large number of receivers, it is not scalable to a large number of senders in terms of memory spaces devoted to store chain commitment for each source; (2) authentication delay peaks during deterministic packet losses (such as long-term network partition and broadcast source failure), which fails to deliver keys and delays acceptance of time-critical information; (3) $ASPBcast$ exacerbates authentication delay further. In TESLA, packets cannot be verified until the source sends authentication keys piggybacked in data packets at later intervals. However, source talks to a random subset of nodes each time and only the nodes being contacted can obtain authentication keys timely. Therefore, packet authentication at noncontact nodes, representing a large body, is postponed and buffer space is consumed to store received and yet unverifiable packets. This delay poses memory-based DoS attacks, lowers data dissemination rate and decreases system responsiveness. Though a separate key broadcast procedure helps push keys fast, it may cause bandwidth contention and complicate protocol management.

In order to lower computational overhead and sustain high data rate in ASPBcast traffic, we adopt TESLA as a building block. But, we overcome the discussed inefficiency, which is due to the fact that *one hash chain is exclusively owned by one identity*. TESLA commitment information for each source must be stored network-wide. A source has to release authentication keys reliably and timely in order to boost its packet authentication at receivers; nobody else can help. *Herein, we propose a better way to define the relationship between hash chain and identity.* The resulting authentication protocol is $ChainFarm$, which largely utilizes the merit of hash chain sharing. The amount of hash chain commitment information stored per node is far less than the number of nodes in the network; and nodes sharing the same hash chain can speedup verification of each others' packets via data packets. In summary, $ChainFarm$ has the following advantages: (1) It is scalable to a large number of senders; (2) It reduces packet authentication delay in deterministic path failure and $ASPBcast$ traffic.

The rest of the paper is organized as follows. Section II describes the network and attacker models, and formalizes the broadcast authentication problem. After a brief review of TESLA as background in Section III, we motivate our ideas and present ChainFarm protocol in Section IV. Section V shows how to configure system parameters to tradeoff conflicting requirements. Section VI discusses the practical issues of $ChainFarm$ and extensions. Section VII evaluates performance of ChainFarm. Following an examination of related work in Section VIII, we conclude in Section IX.

II. Problem Description and Assumptions

We use a slotted probabilistic broadcast model to simplify the discussion of $ASPBcast$ traffic. We consider a network with N nodes $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$. Time is divided into small time slots, called *round*, of length T, indexed from 1. In each round R_j $(j \geq 1)$, node n_i selects a random subset of nodes $\mathcal{S}_{R_j}^{n_i}$ to broadcast messages (Probabilistic reception is modeled as probabilistic sending). $\mathcal{S}_{R_j}^{n_i}$ could be randomly selected from a subset of nodes, uniformly from the whole network or weighted. Fixing the size of contact subset, as selection scope becomes larger and network topology becomes more dynamic, $\mathcal{S}_{R_j}^{n_i}$ becomes more diversified, thus postponing key delivery further. When $|\mathcal{S}_{R_j}^{n_i}| = N$, it is essentially flooding. We assume that broadcast messages from different nodes at different time instances are independent.

Due to delayed authentication in TESLA, packets received from source S at node D in round R_i cannot be verified until S sends authentication keys piggybacked in data packets to D in later round R_j $(j > i)$. The interval from the moment source sending the data packet to the moment destination receiving the key is *authentication delay*. The problem we want to solve is how we can organize the relationship between hash chain and identity for TESLA in collaborative $ASPBcast$ environment, so that (a) broadcast authentication scales to a large number of senders in terms of hash chain commitment storage; (b) authentication delay is reduced, even under network path failure, source failure or when a small number of recipients are randomly selected by the source per round. Next, we will clarify our network model and attacker model.

A. Network Model

Mission critical mobile ad hoc networks are composed of thousands of heterogeneous mobile and portable devices, spanning from low-end sensors to high-end PDAs. They are deployed by the same administrative domain and can establish pairwise trust relationship. They are loosely synchronized. Mobile devices are often deployed in several phases to compensate for node failure, upgrade and labor shift. Therefore, network scale is expected to be dynamic.

B. Attack Model

Operating in open and potentially hostile environment, it is desirable to have security barrier to prevent unauthorized access from outside of the administrative domain and to protect precious communication resources. Attacks can be mitigated by access control because attackers do not know the cryptographic information initially. However, they can modify packets or perform DoS attacks by flooding faked packets. Even worse, via physically capturing mobile devices, attackers may read cryptographic information out of memory and exaggerate their infection by replicating compromised identities or faking new ones without physical compromise. Under cover of faked identities, attackers can perform various attacks, like jamming and virus infection.

III. Background

In order to have a better understanding of $ChainFarm$, let us briefly review how TESLA works and elaborate its inefficiency in authenticating $ASPBcast$ traffic.

A one-way hash chain is a sequence of a fixed-length bit strings $K_0, K_1, \ldots, K_w$ generated via a one-way hash function H, wherein $K_w = H(S), K_{w-1} = H(K_w), \ldots, K_1 = H(K_2), K_0 = H(K_1)$ and S is a bit string of arbitrary length, called seed. A hash chain is uniquely defined by S and H, denoted by $\langle S \rangle$. Any node, knowing K_0, can determine whether a particular value v belongs to hash chain $\langle S \rangle$ or not by recursively applying H over v until a match with K_0 or a failure. K_0 is named commitment key for $\langle S \rangle$. Security of one-way hash chain is established on the fact that it is almost impossible to know inputs given hash results. Here is how TESLA works:

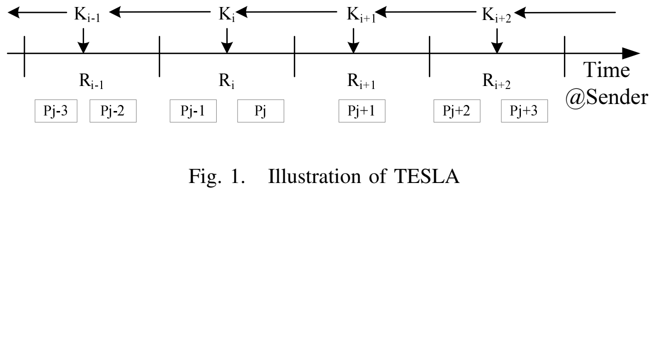

Fig. 1. Illustration of TESLA

The sender splits time into even time intervals called round and generates a one-way hash chain. A hash value is assigned to the corresponding round, K_i to round R_i $(1 \leq i \leq w)$ and K_i is used to sign packets generated in round R_i as shown in Figure 1. Each hash value is released d time intervals after the assigned round. To initialize TESLA, a receiver needs to be loosely synchronized with the sender and know d, commitment key K_0, and H. When sending a message P_j to receivers in round R_i, the sender attaches to P_j a Message Authentication Code (MAC), which is generated based on K_i and packet content. The sender also sends the most recent key it can disclose. When receiving a message P_j sent in R_i, the receiver checks that the sender has not yet reached the round when it discloses key K_i; otherwise, it drops P_j. It then extracts the disclosed key K_{i-d} and check its legitimacy by verifying, for some earlier key $K_v, (v < i - d)$ that $K_v = H^{i-d-v}(K_{i-d})$. If the released key is correct, the receiver buffers P_j, removes all the packets which were sent in and before round R_{i-d} and accepts those with correct MAC.

In the following section, we will assume d has value 1. However, our scheme applies to cases when $d \geq 1$ as well.

In TESLA, a hash chain is exclusively owned by a broadcast source. It suffers from memory and authentication delay issues under $ASPBcast$ traffic. First, a node needs to keep commitment information for each source. In total, $O(N)$ memory is occupied by commitment keys, thus making it unscalable to large network size. Second, probabilistic contact by the source prolongs authentication delay. Suppose a node is contacted by a source in each round with probability p, authentication is delayed by $\frac{1}{p}$ rounds on average. The smaller is p, the worse is the delay. Third, upon network path failure or source chunk, broadcast packets buffered at receivers may never have a chance to be authenticated due to failure of key delivery.

IV. ChainFarm Protocol

Our work is motivated by hash chain sharing. Let us first consider two extreme cases. One extreme is that a hash chain $\langle S \rangle$ is shared by the whole network. Memory for hash chain commitment is $O(1)$. There is zero authentication delay since nodes can use locally stored keys to authenticate packets. However, it lacks the capability of source authentication and any compromised node can claim to be somebody else. Another extreme is TESLA. It supports source authentication and full resilience against compromise because a hash chain is uniquely owned by a sender. However, memory to store chain commitments does not scale and authentication is deferred in $APSBcast$.

Considering the advantages of both extremes, a natural solution is to support an appropriate degree of hash chain sharing to trade memory consumption and authentication delay with resilience against compromise. In $ChainFarm$ protocol, commitment information for hash chains of all senders are predistributed to reduce trust negotiation delay in dynamic networks and a combinatory design is applied to TESLA via chain sharing. Various notations and symbols are summarized in Table I.

TABLE I
Notations and Symbols

$\mathcal{HC}$	A hash chain pool
N	Total number of nodes in the network
$\widetilde{N}$	Maximal allowable number of nodes
$\mathcal{N}$	The set of nodes in the network
α	Number of distinct hash chains in the system $\alpha = \lvert\mathcal{HC}\rvert$
β	Number of hash chains held by each user
$\mathcal{HC}_{ID}$	A set of hash chains held by user with identity ID. $\lvert\mathcal{HC}_{ID}\rvert = \beta$
$k_c(x)$	Expected number of hash chains compromised when x nodes are broken in
M	Maximal memory size for hash chain and commitment key storage
$\mathcal{C}$	Scale factor
D	Expected authentication delay
p	Contact probability

We assume that each node pre-stores its one-way hash chains and all the chain commitment information for other sources before being deployed in field. This is a valid assumption in mission-critical network, such as first responder systems, since all the devices are managed by a single administrative domain or several with security negotiation. Furthermore, because networks typically last for hours or days, we assume that authentication keys in hash chains are never used up using hierarchy hash chain management scheme [9]. This avoids bootstrapping and maintenance of hash chains with senders. Compared with online hash chain negotiation, predistribution is more flexible in scenarios where communication parties continuously change. We will relax this assumption with online hash chain update in Section VI. Different from TESLA, a hash chain pool $\mathcal{HC}$ of α hash chains is constructed for the whole network and nodes store the commitment information for all the hash chains in $\mathcal{HC}$. Each hash chain is shared by several sources and each source S with identity ID has a unique set $\mathcal{HC}_{ID}$ of β hash chains. There is one-to-one mapping from ID to $\mathcal{HC}_{ID}$ so that given ID, $\mathcal{HC}_{ID}$ can be automatically calculated. Because a unique set $\mathcal{HC}_{ID}$ is assigned to each ID, source identity can be verified and impersonation is prevented. All the hash chains have the same releasing schedule. Message signing and verification use all β hash chains associated with senders' identity. If β is 1, $ChainFarm$ is exactly TESLA except chain predistribution. Now we assume cryptographic information cannot be extracted from memory of compromised nodes. This assumption is relaxed in Section VI. Next, we present details of ChainFarm.

A. Hash Chain Predistribution

Each node stores a combination of β hash chains and all the commitment keys for α hash chains. (The actual number of stored commitment keys is $\alpha - \beta$, excluding the hash chains self-owned. However we use α for simplicity.) In order to support source authentication, $\binom{\alpha}{\beta} \geq N$. Otherwise, some nodes must share the same combination. First, a hash chain pool $\mathcal{HC} = \{\langle S_1 \rangle, \langle S_2 \rangle, \ldots, \langle S_\alpha \rangle\}$ is generated and indexed from 1 to α. A set of β hash chains $\mathcal{HC}_{ID} = \{\langle S_1^{ID} \rangle, \langle S_2^{ID} \rangle, \ldots, \langle S_\beta^{ID} \rangle\}$ is assigned to the node with identity ID. Nodes assigned with the same hash chain $\langle S_i \rangle$ compose a hash chain group of $\langle S_i \rangle$. Members of a group help authenticate traffic for each other, since the data packets from a member contain the keys to authenticate prior data packets from other members.

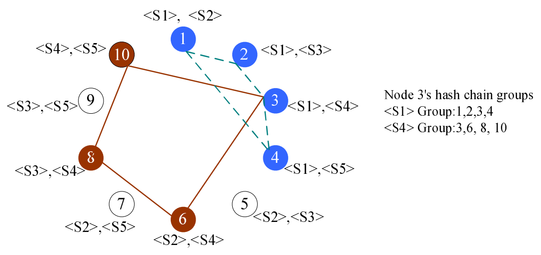

Fig. 2. Hash Chain Assignment

An example of a 10-user network is shown in Figure 2, wherein $\alpha = 5$, and $\beta = 2$. The hash chain pool consists of 5 distinct hash chains $\{\langle S_1\rangle, \langle S_2\rangle, \langle S_3\rangle, \langle S_4\rangle, \langle S_5\rangle\}$ and each user is assigned a unique pair of hash chains and preloaded with 5 commitment keys. For instance, node 1 and node 8 are allocated with $\{\langle S_1\rangle, \langle S_2\rangle\}$ and $\{\langle S_3\rangle, \langle S_4\rangle\}$, separately. The hash chain group for $\langle S_1\rangle$ and $\langle S_4\rangle$ include nodes 1, 2, 3, 4 and nodes 3, 6, 8, 10, respectively. Node 3 belongs to both $\langle S_1\rangle$ and $\langle S_4\rangle$ groups. If any message P from node 3 is received, keys piggybacked in later data packets from group members of $\langle S_1\rangle$ and $\langle S_4\rangle$ authenticate P.

Next, we will show (a) how to allocate a hash chain combination to a node; (b) how to allocate ID so that given an ID, we can infer hash chain indexes in $\mathcal{HC}_{ID}$ uniquely. With this unique binding of ID and hash chains, the source cannot be impersonated. When receiving broadcast packets, a receiver can immediately infer the chain indexes of the sender, without extra two-way communication.

- **Hash chain combination allocation**: For each node, it is allocated a combination of β hash chains which has never been assigned before. This assures source authentication and all the combinations are useful.
- **Unique binding of ID and $\mathcal{HC}_{\mathbf{ID}}$**: After combination assignment, ID is set equal to the concatenation of increasing order of hash chain indexes in $\mathcal{HC}_{ID}$. For example, if node A is assigned with chains $\{\langle S_1\rangle, \langle S_3\rangle, \langle S_5\rangle\}$ ($\beta = 3$), A's ID is $1|3|5$.

Algorithm 1 formalizes the above procedure to assign hash chain combination and bind ID . This algorithm runs offline at trust servers before devices are deployed in field.

Algorithm 1

```
(a) Generate a set $\mathcal{HC}_{\mathcal{COMB}}$ including all the $\binom{\alpha}{\beta}$ combinations of $\beta$ hash chains from $\mathcal{HC}$.
(b) For each node $n$ in $\mathcal{N}$ {
  (1) randomly remove one combination from $\mathcal{HC}_{\mathcal{COMB}}$ and assign it to $n$.
  (2) Load $n$ with $\beta$ hash chains it has been assigned in (1) and commitment keys of all the $\alpha$ hash chains.
  (3) Bind n's ID with its hash chain combination. }
```

B. Broadcast Authentication

Similar to TESLA, time is divided into rounds and keys in each hash chain are associated with the corresponding rounds. All hash chains have the same releasing schedule. *One-way hash signature generation and validation use all the hash chains in* $\mathcal{HC}_{Sender}$. When sending a message P to receivers in round R_i, the sender with ID attaches β MACs to P, one MAC for each hash chain in $\mathcal{HC}_{ID}$. It also includes the most recent keys which it can disclose for each hash chain. A sequence ID is included in "Msg" to prevent replaying attacks. The message format of signed broadcast packet (SigndBrdcst) is as follows:

$$\begin{aligned} SigndBrdcst &= ID, Msg, MACList, ReleasedKeyList \\ MACList &= MAC_1(ID, Msg), .., MAC_\beta(ID, Msg) \end{aligned}$$

When receiving a message P sent in R_i by a sender with ID, the receiver checks that its signing keys are not disclosed yet. It then extracts the disclosed keys for round R_j ($j \leq i-d$), and updates the key commitment information for the corresponding hash chains with legitimate disclosed keys. If some signing keys belong to hash chains owned by the receiver, it verifies the MACs signed by those hash chains immediately. If P has any incorrect MAC, it is dropped; otherwise, it is stored in buffer. Finally, the receiver checks all the buffered packets which were sent in and before round R_j. Packets with at least one incorrect MAC are dropped; receiver accepts all packets *with* β *verifiable and correct MACs*. Importantly, dropping false packets with any incorrect MAC cleans the buffer effectively and prevents buffer overflow with false packets. This is only enabled by the collaborated and fast key delivery property of $ChainFarm$.

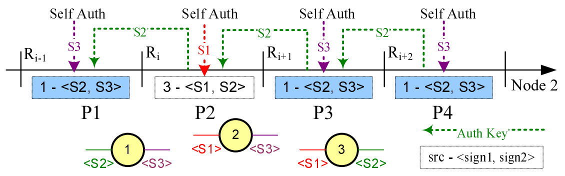

Fig. 3. Authentication via Hash Chain Sharing

The idea of decreasing authentication delay via hash chain sharing is illustrated in Figure 3. Three nodes form a network. Each node randomly selects another node to send a single broadcast packet per round. Let us zoom into Node 2. Packet reception history from round R_{i-1} to round R_{i+2} is labeled. In TESLA, a node has a unique hash chain. Therefore, packets from Node 1 can be authenticated only in later rounds when Node 1 contacts Node 2 again. At Node 2, the authentication delay of packets $P1$ and $P3$ is 2 and 1 interval separately. However, using $ChainFarm$, nodes sharing the same hash chains can boost authentication of broadcast packets from each other. In Figure 3, the hash chains assigned to Node 1, 2, and 3 are $\{\langle S2\rangle, \langle S3\rangle\}$, $\{\langle S1\rangle, \langle S3\rangle\}$, and $\{\langle S1\rangle, \langle S2\rangle\}$ separately. $P1$ is signed by hash chain S_2 and S_3. Since hash chain $\langle S2\rangle$ is shared by Node 1 and Node 3, MACs signed by $\langle S2\rangle$ can be verified when packets from either node arrives; while MACs signed by $\langle S1\rangle$ and $\langle S3\rangle$ can be self-authenticated by Node 2 because it owns these two chains. Authentication delay of $P1$, $P2$ and $P3$ is 1 interval, the optimal value.

V. System Configuration

In this section, we will provide general guideline to configure $ChainFarm$, considering factors of memory, computation and communication overhead, security and delay.

A. Objectives and Their Relationship

System performance depends on the choices of α and β. Generally speaking, we have five performance objectives.

Objective 1 *Memory Efficiency*: Given a maximal network size $\widetilde{N}$, we need to find a *Hash Chain Pool* $\mathcal{HC}$ and a *hash chain allocation* to restrict the memory consumption per node $\alpha+\beta*\mathcal{C}$ within the maximal memory space of M units and still enable source authentication, that is $\binom{\alpha}{\beta} \geq \widetilde{N}$. Importantly, we use $\widetilde{N}$ rather than N to allow dynamic node addition. A memory unit is defined as the amount of space to store a commitment key. And the space to store a hash chain is usually greater than 1 unit, denoted by a scale factor $\mathcal{C}$. $\mathcal{C}$ varies to trade computation with storage overhead. Applicable data structures are Merkle tree [10], and multi-level Merkle tree [7].

Objective 2 *Computational Complexity*: It is better to use a small number of hash chains to sign outgoing messages and to verify incoming messages, thus with reduced β.

Objective 3 *Communication Overhead*: Considering valuable bandwidth of wireless networks, we want smaller packet size of fewer MACs per packet, thus with reduced β.

Objective 4 *Resilience Requirement*: Due to cheap manufacture cost, mobile devices can be compromised and the related hash chain information can be read out by attackers easily. Since hash chains are shared among the hash chain group, compromise of several nodes could release hash chains owned by non-compromised nodes. Every compromise will possibly trigger an identity revocation, sometimes leading to expensive system reconfiguration. Therefore, we want to limit the impact of compromise. One metrics to evaluate the resilience against compromise is the percentage of broadcast sources whose hash chain combination is still secret when attackers randomly break into x nodes. This percentage is

$$1-\binom{k_c(x)}{\beta}\Big/\binom{\alpha}{\beta} \tag{1}$$

, where $k_c(x)$ is the expected number of hash chains known by attackers when x nodes are broken in and $\binom{k_c(x)}{\beta}$ is the average number of distinctive hash chain combinations known to attackers when adversaries break into x nodes.

$$k_c(x) = \alpha(1-(1-\frac{\beta}{\alpha})^x) \quad [11] \tag{2}$$

Intuitively, larger α and smaller β improve resilience against compromise because less information is shared.

Objective 5 *Authentication Delay*: Packet authentication is delayed to the moment when all β keys are collected. We want to increase the degree of hash chain sharing so that members of hash chain group could help authenticate each other's packets, which prefers larger β and smaller α.

B. Parameter Configuration

It is hard to optimize all the objectives simultaneously and users often have expected performance in mind, such as the maximal network size and authentication delay. Hence, we transform the configuration problem into the following optimization problem: for a given statistical traffic model, the maximal network size $\widetilde{N}$, expected authentication delay D and maximal memory consumption M, we maximize the resilience against compromise. We only consider β from 2 to 7 to avoid excessive communicational and computational overhead of β MACs. Resilience requirement in Equation (1) is approximated by $\frac{\alpha}{\beta}$ to remove the effect of x, which is the number of compromised nodes.

$$\begin{aligned} \max \quad & resilience \approx \frac{\alpha}{\beta} \\ s.t. \quad & \alpha+\beta*\mathcal{C} \leq M \\ & \overrightarrow{E}[delay] \leq D \\ & \binom{\alpha}{\beta} \geq \widetilde{N} \\ & \beta = 2..7 \end{aligned} \tag{3}$$

Authentication delay is a function of T, the length of rounds. Different T incurs different values for absolute delay. For consistent measurement, the unit of delay is defined as rounds.

A heuristic algorithm to solve (3) is presented in Algorithm 2. We assume the expected delay $\overrightarrow{E}[delay]$ is known under a certain traffic model. We iterate β from 2 to 7 and find the corresponding α value, reduced from the maximal α, which is computed via the memory constraint, until the point when the expected delay is at most D under the condition of source authentication. We record an α for each β, which has the best resilience and satisfies both delay and source authentication requirements among all αs for a β. Finally, we locate β^* and α^* with maximal resilience against compromise, $\frac{\alpha}{\beta}$, while satisfying all the constraints.

Algorithm 2

```
for $\beta = 2$ to 7 {
    $\alpha = M - \mathcal{C} * \beta$; calculate $\overrightarrow{E}[delay]$;
    while (($C(\alpha,\beta) > \widetilde{N}$) & ($\overrightarrow{E}[delay] > D$)) {
        $\alpha = \alpha - 1$;  calculate $\overrightarrow{E}[delay]$);}
    if ($C(\alpha,\beta) < \widetilde{N}$) $\alpha = \alpha + 1$;
    record $\alpha$ for the current iteration of $\beta$;
}
pickup $\beta^*$ and corresponding $\alpha^*$ so that $\frac{\alpha}{\beta}$ is maximized
and all the constraints in (3) are satisfied.
```

The challenge left is how to calculate $\overrightarrow{E}[delay]$. If we have a statistical traffic model, the calculation is easy. Here we introduce *uniform-p traffic model*: in each round, node n_i sends a packet to random $N \cdot p$ nodes uniformly selected from $\mathcal{N}$. p is called contact probability. A node could be selected multiple times by a source in a round. Put in another way, in each round, $N \cdot p$ nodes are about to contact node A and this random set of nodes is uniformly sampled from $\mathcal{N}$. Authentication delay for packet P is the expected time when node A collects all β authentication keys for P. This key collecting problem can be approximated by the coupon collector's problem [12]: out of α types of coupons, one coupon is picked randomly at each trial. How many trials one has to perform before picking all the β coupons? We make the following simplified assumptions: (1) in each trial one coupon is independently chosen; (2) keys collected in the same round as data packet P counts as long as they are received after P; (3) we ignore the effect of self-authentication. The expected number of trials is $\sum_{i=1}^{\beta} \frac{\alpha}{i}$. The expected number of rounds is calculated by dividing the expected trial number by $N \cdot p \cdot \beta$, which is the expected number of trials per round. This yields,

$$E[Delay] = (\sum_{i=1}^{\beta} \frac{\alpha}{i}) * \frac{1}{N \cdot p \cdot \beta} \quad (4)$$

The standard deviation of the authentication delay is

$$sd[Delay] = \sqrt{\sum_{i=1}^{\beta} \frac{1 - i/\alpha}{(\frac{i}{\alpha})^2} * \frac{1}{N \cdot p \cdot \beta}} \quad (5)$$

Unfortunately, the standard deviation is $O(\alpha)$. We need a high confidence that the expected authentication delay is close to the required value. There are two ways to increase the confidence level. One way is to adjust the estimated delay by half of the standard deviation

$$E_{upper}[delay] = E[Delay] + sd[Delay] \cdot 0.5 \quad (6)$$

The other way is to get a stronger bound. Let random variable X denote the number of trials for collecting each type of coupon. We desire to find the largest value of $\widehat{m}$ so that

$$1 - Pr[X > \widehat{m}] \leq prob \quad (7)$$

, wherein $prob$ is the desired probability of the event that the number of trials falls below $\widehat{m}$. ($prob$ is 0.8 in simulation.) Let Z_i^r denote the event that i_{th} key was not picked in the first r trials. Clearly, $Pr[Z_i^r] = (1 - \frac{1}{\alpha})^r$ and $Pr[X > m] = Pr[\bigcup_i Z_i^m]$. By inclusion-exclusion principle, we have

$$\begin{aligned} Pr[X > m] &= Pr[\bigcup_i Z_i^m] \\ &= \sum_{i=1}^{\beta} (-1)^{i+1} \cdot \binom{\beta}{i} \cdot (1 - \frac{i}{\alpha})^m \quad (8) \end{aligned}$$

The expected delay $\overrightarrow{E}[delay]$ in (3) takes the average of two delays so as to get a good and yet tight expectation.

$$\overrightarrow{E}[delay] = (\frac{\widehat{m}}{N \cdot p \cdot \beta} + E_{upper}[delay])/2 \quad (9)$$

VI. Extension and Discussion

In this section, we will discuss online hash chain update and several practical issues of $ChainFarm$.

(1) Online hash chain update: ChainFarm uses hash chain predistribution so that members of a hash chain group are assured to own the same hash chains. We can use the multi-level TESLA proposed in [9] to prolong hash chain lifetime. But it is still desirable to update hash chains online when impersonation is detected, when TESLA keys are used up, or during system reconfiguration. A centralized approach can be used. Whenever online update is needed, a trusted server broadcasts signed commitment information for all hash chains in the newly generated chain pool and sends the encrypted hash chain combination for each node.

(2) Compromise detection: If everyone runs $ChainFarm$ honestly, nobody can be impersonated. If cryptographic information cannot be extracted from memory of compromised nodes, nodes other than the compromised ones cannot be impersonated either. However, due to the nature of hash chain sharing, several compromised nodes may collude with each other and infer private hash chain combinations of other uncompromised nodes. For example in Figure 3, if attackers compromise Node 1 and 2, they know the hash chains owned by Node 3, which are supposed to be secret. In this way, attackers can pretend to be an existing device or even a nonexistence device. Impersonation of nonexistent devices is possible because system is usually designed for scalability. Some hash chain combinations are preserved for node addition.

We use the distributed replication detection techniques proposed in [13] to prevent replication or impersonation after compromise. Contrary to hash chains which are shared among several nodes, a unique private key is given to a node. Hence, it is impossible for attackers to know other uncompromised devices' private keys. The identity-based public key system is implemented to save transmission overhead for certificate or public key, wherein one can calculate node ID's public key as $f(ID)$. ID associated with public-key cryptography may be different from the ID used in $ChainFarm$. We will first summarize how the replication detection protocol works in 4 steps and then explain how it applies for $ChainFarm$.

1. Location Claim: Each node periodically broadcasts its location claim composed of ID and current position along with a public-key signature.

2. Neighborhood Forwarding A neighboring node v has a probability to forward a *valid* claim to a subset of randomly selected g witness locations using GPSR routing protocol [14]. A claim is valid if its signature is authentic and claimed location is plausible (location claimer is in v's transmission range). Any node failing to send valid location claim is blacklisted by its neighbors.

3. Witness Duplication Check: After a node closest to a witness location receives a location claim, it verifies the signature and check the ID against all the valid location claims they have already received. Any ID with more than two different location claims in a claim period is rendered as a replicated identity and triggers revocation.

4. Network-wide Revocation: A network-wide broadcast is initiated by the nodes detecting duplication. Broadcast messages include the conflicting claims of the same ID.

Now let us analyze how this protocol helps detect compromise. There are three cases. (a) If attackers impersonate a source without physical compromise, collusion for example, attackers do not know the source's private key. They are unable to generate claims with valid signature and are detected in step 2. (b) If attackers compromise a source and replicate its cryptographic information into several malicious devices, it is exactly replication attack. The malicious ID is detected in step 3. (c) If attackers only compromise a source without replication, it is not detectable. However, attackers' infection is limited to the compromised node. Yet other techniques of behavior monitoring and analysis can detect intrusion if compromised sources act maliciously.

After detection, administrators can be notified. Any node, which confirms the compromised identity ID, can blacklist this ID and cut off any communication associated with it.

(3) Are the increased packet size and computational overhead for extra signatures BAD? We believe the overhead is acceptable. First, MAC signatures are usually generated by secure hash functions, such as SHA-1 and MD5. Typical hash size is 16 bytes for MD5 and 20 bytes for SHA-1, regardless of the message size. Trimming of hash value can further reduce the hash size to 10 bytes. In $ChainFarm$, β is from 2 to 7. While experiments shows that β is 2 or 3 in normal cases, the size of extra $2(\beta - 1)$ released keys and signatures is small, compared with the maximal packet size. We show the typical message overhead in Table II. Second,

TABLE II
MESSAGE OVERHEAD: MAC AND KEY SIZE (BYTE)

Hash Function	$\beta = 2$	$\beta = 3$	$\beta = 4$
MD5 (16B)	32	64	96
SHA-1 (20B)	40	80	120

computation is not a problem since one-way hash computation is light-weighted. It is better to trade computation and packet size for reduced authentication delay and early drop of faked packets.

(4) Is loose synchronization in TESLA hard to achieve? It has been shown that loose synchronization can be securely achieved in both centralized and distributed ways with high accuracy [15][16][17].

(5) Dynamic node addition: By using the maximal network size $\widetilde{N}$ instead of N, we can easily accommodate node addition. After the initial node deployment, the trust server will reserve the rest unused hash chain combinations for additional nodes. When the new nodes arrive, they first contact the offline trust server, loading the assigned hash chain combination and commitment keys for all the hash chains in chain pool and obtaining their ID. Then they can communication with other devices in field freely.

VII. EVALUATION

We adopt *uniform-p traffic model* for system configuration and protocol performance evaluation. First, we investigate the typical network configuration under different network sizes, contact probabilities and required authentication delay with *uniform-p traffic model*. Then we evaluate $ChainFarm$ performance in terms of memory consumption, resilience against compromise and authentication delay. Delay is measured in a simulated environment under a more general *locality-p traffic model*.

A. Configuration of α and β

First, we show configurations of α and β, varying contact probability p, network size $\widetilde{N}$ and authentication delay D.

In Figure 4, we show the setting of α and β in a network composed of 10000 nodes. Each node has 500 units of memory for hash chain storage and $\mathcal{C} = 30$. The expected authentication delay is 1.67 rounds. Contact probability varies from 0.001 to 0.015. β is labeled in the transition points of the curve as b. When the contact probability increases, α increases and β decreases since more nodes are contacting each other and consequently less sharing is required to achieve the same authentication delay.

In Figure 5, we study the typical configuration of α and β for different authentication delay requirements and network sizes with $M = 400$ and $\mathcal{C} = 40$. Network size varies from 1000 to 10000 and expected delay changes from 1.4 to 2.0 intervals. Each upper slope in the 5 curves indicates a change of β from 2 to 3. When the required delay becomes less strict, larger α is realized. For a small network size, our scheme prefers a small β of 2. When the network size becomes larger, β increases to 3 so as to support source authentication. Because we have a constant contact number ($N * p = 50$) per round, α and β remain constant if they can support the network size, depicted as the horizontal tails in each curve. In this way, our scheme can easily accommodate node addition without change to α and β.

B. Memory Consumption

$ChainFarm$ protocol keeps memory consumption below M; while in TESLA, it is $N+\mathcal{C}$, with N commitment keys and 1 hash chain. For the configuration in Figure 5, the memory consumption for $ChainFarm$ is below or equal to 400 units; while in TESLA it is 1040 for 1000 nodes and 10040 for 10000 nodes. For a larger network scale, the memory saving is more dramatic. With $M = 400$ and $C = 40$, $\beta = 3$ supports 10000 nodes in Figure 5. ChainFarm can easily accommodate 50 billion nodes with $\beta = 7$.

C. Resilience against Compromise

In Figure 6, we evaluate the resilience against compromise by examining the percentage of affected broadcast senders, (1− Eq.(1)), under the configuration in Figure 4. When the contact probability increases, less sharing is required in order to achieve the same authentication delay. This leads to better resilience against compromise. As more collusive attackers appear, the percentage of senders who are affected increases. However, the increased rate is low for high contact probability. For $p = 0.009$, 10 compromised nodes only affect 0.35% nodes; and 1.05% nodes for $p = 0.006$. Abnormally, the resilience for $p = 0.01$ is worse than $p = 0.006$. The reason is that though the absolute number of affected broadcast senders is small for $p = 0.01$, the maximal allowable network size is also smaller: $\binom{155}{2}$ for p=0.01 v.s. $\binom{119}{3}$ for p=0.006.

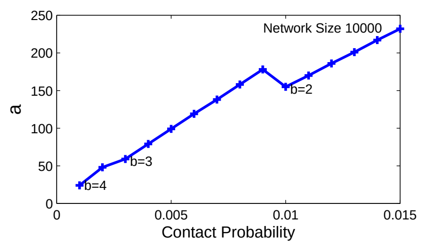

Fig. 4. Configuration of α and β

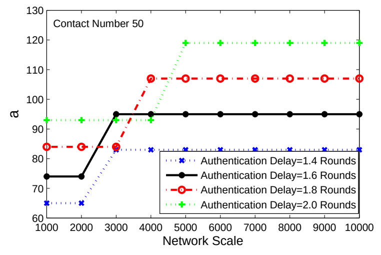

Fig. 5. Configuration of α and β

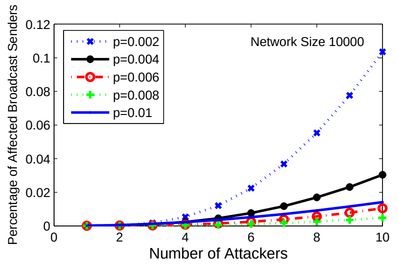

Fig. 6. Resilience against Compromise

Unfortunately, strategic attackers may compromise α/β nodes which have distinct hash chain combinations so as to demolish the secrecy of the whole network. However, it takes time to compromise devices and ChainFarm optimizes the resilience against compromise via choice of α and β. Upon substantial compromise, ChainFarm can reconfigure using online hash chain update. For most cases, it is better to trade compromise resilience for improved authentication delay and early dropping of faked packets.

D. Authentication Delay

Now, we study how the authentication delay looks like in simulation and what is the gap between the actual authentication delay and required one when systems are configured under *uniform-p traffic model*. We test both $ChainFarm$ and $TESLA$ under a more general *locality-p traffic model*, which takes the inaccuracy of traffic modeling into account.

In *uniform-p traffic model*, each node chooses $N \cdot p$ random nodes uniformly from the whole network and sends one packet to each of them per round. Whereas, some applications may have locality preference, such as in wireless networks. Therefore, we have adopted the small world model [18] in *locality-p traffic model* for evaluation. For any node v other than A, it is chosen by node A with probability proportional to $d(A,v)^{-\gamma}$, wherein $d(A,v)$ is the hop distance from node v to A. The hop distance is calculated in a 2-dimensional attribute space with nodes randomly deployed. $d(A,v)^{-\gamma}$ is divided by a normalizing constant $\sum_{v\in\mathcal{N},v\neq A} d(A,v)^{-\gamma}$ to obtain a probability distribution. If $\gamma = 0$, *locality-p traffic model* equals to *uniform-p traffic model*. Increasing γ stresses the preference over "nearby" neighbors.

In the first set of experiments, we compare the authentication delay of $ChainFarm$ with $TESLA$. The network includes 100 (200) nodes with $p = 0.1$, $M = 100$ and $\mathcal{C} = 20$. The resulting configuration is $\alpha = 16$ and $\beta = 2$ ($\alpha = 31$ and $\beta = 2$) when the required authentication delay is $\frac{5}{3}$ rounds. Each simulation instance lasts for 100 rounds. It is possible that some packets are never authenticated due to probabilistic contacts by their sources. We have separated the number of unauthenticated packets from average delay measurement. The mean node degree is fixed at 9.

Figure 7 shows that the authentication delay of $ChainFarm$ matches the required delay of $\frac{5}{3}$ intervals, no matter how γ changes. While for $TESLA$, the delay is much higher. It is smaller than the expected value of 10, given $p = 0.1$, because packets, which are unable to be authenticated, are counted separately. As we increase γ, the delay of $TESLA$ decreases since locality increases the contact probability over a small subset of nodes.

The same effects are observed on the number of unauthenticated packets in Figure 8. We believe that $TESLA$ will be much worse in terms of number of unauthenticated packets because of packet timeout policy. In our experiments, we allow infinite waiting period till the end of simulation. While in practice, if the packets cannot be authenticated in a predetermined period, receivers will drop them.

In the second set of experiments, we show the impact of contact probability on authentication delay. The network setting is 1000 nodes with $M = 200$ and $\mathcal{C} = 30$. Under the condition that required authentication delay is $\frac{5}{3}$ rounds, we have plotted the average authentication delay in Figure 9, varying the contact probability from 0.01 to 0.10 and γ from 0 to 4. Configuration of α and β is shown in Table III.

TABLE III
PARAMETER CONFIGURATION

Contact Probability	α	β	Contact Probability	α	β
0.01	20	3	0.06	93	2
0.02	39	3	0.07	108	2
0.03	46	2	0.08	124	2
0.04	62	2	0.09	139	2
0.05	77	2	0.10	140	2

Figure 9 proves that the actual authentication delay matches the expected one quite well in a wide range of contact probability except over-estimation at two extremes. For small p, gap between the actual authentication delay and required one is 0.05, which is generally acceptable. This is caused by the fact that we skip the effect of self-authentication in analysis. Small p prefers a large number of hash chains held per node. Therefore, the self-authentication probability increases, decreasing authentication delay. While for the contact probability larger than 0.09, in theory, α should increase as p increases; however it is bounded by constraint of memory size and remains the same no matter how p increases as shown in Table III. Therefore, we see a drop in the authentication delay for $p = 0.10$. Increased γ decreases the performance of $ChainFarm$ protocol, but still around the required authentication delay. Remote nodes, which are occasionally contacted, are difficult to be contacted again in future, thus increasing the overall delay.

We also examine the authentication delay of $ChainFarm$ and $TESLA$ for different contact probabilities under packet

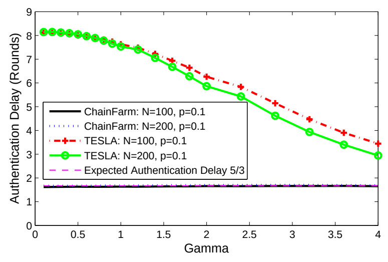

Fig. 7. Authentication Delay (Rounds)

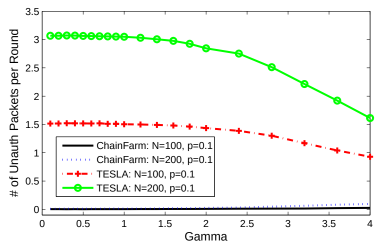

Fig. 8. # Unauthenticated Packets Per Round

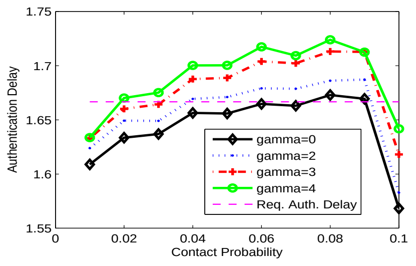

Fig. 9. Authentication Delay (Rounds)

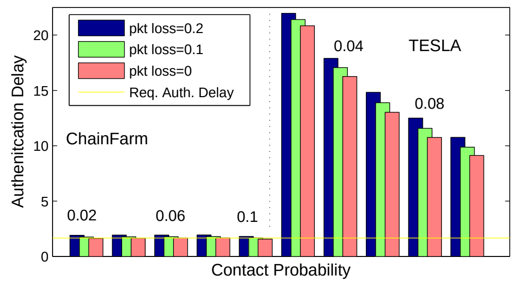

Fig. 10. Authentication Delay under Packet Loss

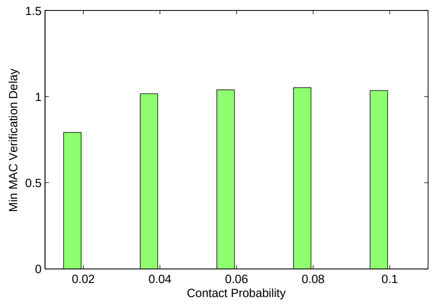

Fig. 11. Minimal MAC Verification Delay

loss in Figure 10. The same configuration as in Table III is used. Packet loss ratio varies among 0.2, 0.1 and 0 (no packet loss). Contact probability changes from 0.02 to 0.1 with 0.02 stepsize. Performance of $ChainAFarm$ and $TESLA$ is shown in the left and right panel, respectively. As packet loss ratio increases, authentication delay increases for both protocols since the effective contact probability decreases and receivers wait for longer periods for authentication keys. However, delay performance of $ChaimFarm$ decays slowly, still around the required authentication delay of 5/3 rounds. This deviation from required delay is caused by inaccurate traffic model. As we can see, authentication delay of TESLA is generally lower than $\frac{1}{p}$ rounds because we separate unauthenticated packets in this measurement. The average number of unauthenticated packets per round for TESLA is more than 7. On contrary, it is less than 0.1 for $ChainFarm$ under all the contact probabilities and loss ratios examined above. We conclude that the coordinated key delivery mechanism in $ChainFarm$ makes authentication delay tolerant of packet losses. Not shown here, another improvement is that $ChainFarm$ enables authentication even under path or source failure; while $TESLA$ does not.

E. Minimal MAC Verification Delay

In this section, we study the minimal MAC verification delay, which is the delay when the receiver of a packet P receives the first authentication key, out of β keys. If the receiver owns a hash chain used to sign P, this delay is 0. Minimal MAC verification delay is an important metrics to evaluate resilience against false packet injection attacks. It is the upper delay bound for false packets to be cleaned out from buffer. Short delay due to partial authentication makes more buffer space devoted to probably authentic packets. Shown in Figure 11, the delay for $ChainFarm$ is below 1.5 rounds. The minimal MAC verification delay for $p = 0.02$ is less than 1 round due to large degree of self-authentication as explained in Figure 9. While, the delay for TESLA is estimated to be $1/p$, from 50 to 10 rounds. In $ChainFarm$, whenever a false MAC signature is detected, the packet is dropped. This early dropping mitigates memory-based DoS attacks, which is achievable via “collaborative” key delivery mechanism in $ChainFarm$.

VIII. Related Work

Our ideas are motivated by SMOCK [11] for public-key management, wherein a unique combination of keys is associated with an identity, rather than one key in public key scheme. However, public key based authentication is too computationally heavy for broadcast authentication at mobile devices. We are interested in light-weighted cryptographic primitives with small authentication delay. Furthermore, no mechanism is designed to detect compromise.

The ideas of $ChainFarm$ to improve authentication delay can be illuminated by Resilient Overlay Networks [19] in P2P domain. The basic principle in RON is that nodes recover from path outages and degraded performance by coordination among overlaid RON nodes, besides the network routing recovery process controlled by core routers. The overlay coordination triggers more timely response to deal with path outage and degraded performance. This principle promotes us to design ChainFarm protocol to coordinate devices to recover from path or source failure and to decrease authentication delay in probabilistic broadcast.

[20] develops several techniques to support multiple senders for the parameter distribution phase. However, it only assumes a small number of senders or a few active senders staying in the network at one time. In presence of a mass of active

senders, the hash chain commitments for each sender should be unfolded, making the memory consumption unscalable for cooperative environment. In addition, it does not consider probabilistic broadcast. [7] proposes a variation of TESLA to support light-weight broadcast authentication in sensor networks. [9] proposes several variations of TESLA to support both long hash chain lifetime and short authentication delay via multi-level TESLA to defend against replaying/DoS attacks. The multi-level TESLA idea in [9] does not aim at $ASPBcast$ traffic; however, it can be integrated with $ChainFarm$ to prolong lifetime of hash chain and shorten length of round. [21] secures one-hop transmission against unauthorized resource access and overhearing via symmetric key scheme, which only fits for unicast traffic in static networks.

IX. Conclusion

In this paper, we identify an important traffic type, "High-rate Any Source Probabilistic Broadcast", in collaborative environment. We propose $ChainFarm$ protocol, which is scalable to a large number of senders, reduces packet authentication delay in deterministic network path failure and probabilistic broadcasting and supports dynamic node addition. The simulation verifies the feasibility of our scheme; and memory and authentication delay is dramatically improved compared with tradition TESLA.

We envision many applications for $ChainFarm$, such as traffic authentication, secure location services and pub-sub systems with dynamic network topology and traffic, whose pattern seen by each node resembles $ASPBcast$ traffic. As future work, we plan to test $ChainFarm$ performance for those applications and work on adaption issues, like distributed online hash chain update and location-aware chain group management to lessen the impact of compromise.

Acknowledgment

This research is supported by Boeing Trusted Software Center at the Information Trust Institute, University of Illinois.

References

[1] M.-J. Lin, K. Marzullo, and S. Masini, "Gossip versus deterministic flooding: Low message overhead and high reliability for broadcasting on small networks."

[2] L. Rodrigues, S. Handurukande, J. Pereira, R. Guerraoui, and A. Kermarrec, "Adaptive gossip-based broadcast," in *DSN'03*.

[3] F. Bai and A. Helmy, "Impact of mobility on mobility-assisted information diffusion (maid) protocols," USC, Tech. Rep., 2005.

[4] H. H. Y. Das, S.M. Pucha, "Performance comparison of scalable location services for geographic ad hoc routing," in *INFOCOM*, 2005.

[5] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," *Proceedings of IEEE Symposium on Security and Privacy'05*, vol. 00, pp. 49–63, 2005.

[6] A. Perrig, R. Canetti, D. Tygar, and D. Song, "The tesla broadcast authentication protocol," 2002.

[7] A. Perrig, J. D. Tygar, D. Song, and R. Canetti, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of IEEE Symposium on Security and Privacy'00*. IEEE Computer Society, 2000, p. 56.

[8] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *CCS '03*, 2003.

[9] D. Liu and P. Ning, "Multi-level μ tesla: Broadcast authentication for distributed sensor networks," in *TECS*, 2004.

[10] R. C. Merkle, "Secrecy, authentication, and public key systems," Ph.D. dissertation, Dept. of Electrical Engineering, Stanford Univ.

[11] W. He, Y. Huang, K. Nahrstedt, and W. C. Lee, "Smock: A self-contained public key management scheme for mission-critical wireless ad hoc networks," *Percom*, 2007.

[12] S. Har-Peled, "The stable marriage problem, and the coupon collectors problem," *Lecture Notes*.

[13] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, 2005.

[14] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Mobile Computing and Networking*, 2000, pp. 243–254.

[15] R. R. R. Rowe, A. Mangharam, "Rt-link: A time-synchronized link protocol for energy- constrained multi-hop wireless networks," in *SECON*, 2006.

[16] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, 2002.

[17] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," 2005. [Online]. Available: citeseer.ist.psu.edu/article/song05attackresilient.html

[18] J. Kleinberg, "The small-world phenomenon: an algorithm perspective," in *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000.

[19] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Symposium on Operating Systems Principles*, 2001.

[20] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks," in *MOBIQUITOUS '05*, 2005.

[21] L. Eschenauer and V. Gligor, "A key management scheme for distributed sensor networks," in *IEEE Symposium on Security and Privacy*, 2002.