TECHNICAL RESEARCH REPORT

A Certificate-based Light-weight Authentication Algorithm For Resource-constrained Devices

by Ayan Roy-Chowdhury, John S. Baras

CSHCN TR 2005-4
(ISR TR 2005-83)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

A Certificate-based Light-weight Authentication Algorithm For Resource-constrained Devices

Ayan Roy-Chowdhury, John S. Baras
Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland
College Park, Maryland 20742
Email: {ayan, baras}@isr.umd.edu

***Abstract*— In this work, we analyze and extend a recently proposed design of digital certificates called TESLA certificates. Certificates are a necessary tool in today's secure networks to certify the identity of nodes taking part in communication. Most prevalent certificate technologies make use of public-key cryptography. Messages generated by the user are signed using its private key, and the signature can be verified by any node who knows the user's public key via its certificate. Signature generation and verification using public-key cryptography is computationally expensive for devices with limited computation power and energy resources. In this situation TESLA certificates can be very useful to certify identity, since they rely on symmetric cryptography which is computationally much more efficient. In this paper we explain the concept of TESLA certificates and provide a preliminary description of proposed modifications to the original algorithm to strengthen its security. We extend the original proposal by combining hash chains with TESLA certificates and come up with an efficient source and message authentication protocol based on symmetric key certificates. We also propose a new type of TESLA certificates called Group Certificates for use in multicast group communication. Through analysis, we show that our protocol is secure against malicious adversaries. We also give an initial estimate of the performance of our algorithm and the related comparison to public-key signatures, and we highlight network scenarios where the TESLA certificates could be particularly useful.**

Keywords: Source authentication, Message Authentication Code (MAC), digital certificate, public-key cryptography, TESLA, hash chain.

I. Introduction

Authentication is an essential building block in enabling secure communication amongst a group of nodes. Some important reasons for authentication include the following.

- Source authentication guarantees receivers that a particular application message was originated by the node indicated as the source in the message.
- Authentication is useful for message integrity checks - to make sure that a message has not undergone unauthorized modifications while in transit from the source to the destination(s). In this case authorized nodes append certain immutable fields to the message, in a process similar to entity authentication, such that message tampering violate the integrity of the immutable fields and is easily detected.
- Authentication can be used to verify the identity of nodes in order to allow them access to certain services/applications in the network (access control).
- Authentication is necessary to establish a secure routing path from the source to the destination, where the routing control messages between any two pair of nodes (or between the source and the destination) need to be verified as coming from legitimate nodes in the path. This is important in ad hoc networks, where intermediate nodes that are not trusted act as routers.

The most light-weight form of authentication (and associated message integrity check) is possible when two communicationg nodes A and B share a secret exclusively between themselves and make use of this secret, or a key derived thereof, to "sign" the messages between themselves. Since no other node knows the secret, A can be assured that the message originated at B, and vice versa. The secret or key used can be based on symmetric cryptography that is fast, efficient, and does not consume significant computation or energy resources at the communicating nodes. The corresponding message signature is usually a Message Authentication Code, or MAC in short (for example, HMAC [1]), which is resource-efficient to compute and to verify, and limited in size.

However, when multiple parties are taking part in a communication session, a shared secret between the parties is not a solution. Since everyone knows the same secret, it is impossible for the receivers to know for sure whether the message originated at the alleged source, or was it spoofed by another node in the group that knows the secret. At best, a shared secret in this setting can assure the involved nodes that the message originated from someone within the group (assuming the secret has not been leaked to outsiders). It is also possible that all the nodes do not share secret beforehand, for example when a group of nodes with no prior knowledge of one another take part in a communication session, and erase the security information once the session terminates (this is true even for two nodes with no prior history together). In this situation authentication is done based on asymmetric techniques where each node possesses a unique secret known to no other node, and makes use of that secret to authenticate itself, or the messages it generates. Public key cryptography

allows such asymmetric authentication to take place. In public-key cryptography, each source uses its private key to sign messages it generates, creating a digital signature that is appended to the message [2]. The receivers can verify the signature using the corresponding public key of the node, which is known to everyone from the source's certificate. The primary requirement is that all users have access to a common third party node called the Certificate Authority (CA) that is universally trusted. The CA is responsible for binding a node's identity to its public key in the node's public-key certificate - for example, PGP [3] and X.509 [4], which are the two most commonly used certificate formats. The certificate can be freely distributed to all nodes in a network, and the correctness of the certificate is verifiable by any node that has access to the CA. Apart from facilitating secure authentication of nodes and message integrity checks, public-key cryptography also provides *non-repudiation* - a node cannot deny later that it generated a message that has been signed using its private key.

Public-key cryptography is a powerful tool that facilitates authentication, message integrity and also data encryption. However, it is computationally very expensive (both in CPU cycles and energy expenditure) to generate digital signatures for messages, and also to verify them. The public and private keys are larger in size compared to symmetric keys, and the certificates also take up considerable storage space. In wireless networks where many of the nodes might have resource constraints, public-key cryptography can be a severe burden. For example, handheld devices have limited processor power, storage capacity and available energy. Performing digital signature generation and verification frequently can consume significant processor capacity and drain the battery quickly. The issue has gained importance due to the proliferation of wireless networks of mobile resource-limited nodes and wireless sensor networks consisting of tiny sensor nodes with severe resource constraints. Therefore, research efforts are underway to design algorithms that allow secure authentication and message integrity for such devices without relying on public-key cryptography.

In this work, we focus on the above problem of authentication for resource-constrained devices. Several novel authentication mechanisms have been proposed, some of which attempt to mitigate the resource expenditure in public-key based digital signatures, while others propose new approaches [5]–[12]. A new type of certificate for authentication of compute-constrained devices has been proposed in [13]. Named TESLA Certificate, it is based on the TESLA broadcast authentication protocol [14]. TESLA certificate is based on symmetric cryptographic primitives - MAC computations using keyed hash functions - and uses delayed disclosure of the key by the CA, to achieve the asymmetry required for authentication in group communication. Due to the use of MACs to generate and verify certificates, the scheme is well-suited to mobile nodes with limited resources. We make use of the TESLA certificate concept to develop new authentication tools. In particular, the work presented in [13] and the related paper

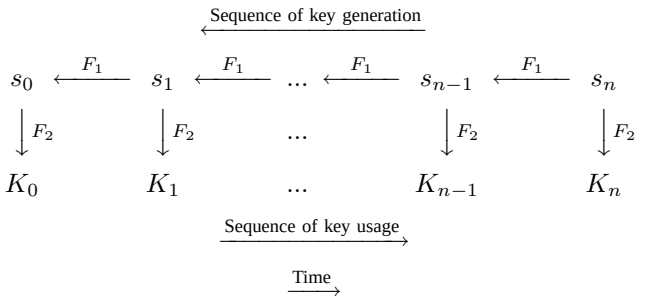

Fig. 1. TESLA key generation

[15] has several security weaknesses and is limited in scope. We attempt to fix these weaknesses and extend the algorithm for entity authentication and message integrity for resource-constrained devices.

The rest of this paper is organized as follows. In section II, we briefly review the TESLA authentication protocol. TESLA certificate and its proposed application to a hybrid network topology is described in III, alongwith an analysis of the strengths and weaknesses of the proposal. We describe our modifications to correct the weaknesses in the original TESLA certificate proposal in section IV. We extend the scope of TESLA certificates by adding more features described in sectionV. We discuss the security of our modifications/additions in section VI. A preliminary analysis of the performance of our extended TESLA certificate algorithms is given in section VII, alongwith highlights of some issues involved. Section VIII outlines our current research efforts for TESLA certificates. We conclude the paper in section IX.

II. Review of TESLA Authentication Protocol

The TESLA broadcast authentication protocol [14], [16] represents a fundamental paradigm shift in source authentication in a group setting. TESLA achieves asymmetric authentication between a source and receivers through the use of symmetric cryptographic MAC functions. The asymmetry is obtained through the *delayed disclosure* of the authentication keys. We give a brief description of TESLA in the following paragraphs.

TESLA divides the time of transmission by the source into n intervals of equal duration. The source generates a random key seed s_n for interval n, and computes a one-way hash chain by repeatedly applying a one-way function F_1 to s_n. The number of elements of the hash chain correspond to the number of intervals that the source transmits. The source computes the MAC computation key for each time interval by applying a second one-way function F_2 to each element of the hash chain. The functions F_1, F_2 are publicly-available and known to all the receivers. The algorithm is illustrated in fig. 1.

The sender uses the keys in the reverse order of their generation, that is, starting with K_1 in interval 1, followed by K_2 in interval 2, and so on. Owing to the one-way property of F_1 and F_2, it is computationally infeasible for any node to generate s_i knowing K_i, or to generate s_{i+1} knowing s_i.

The sender bootstraps the hash chain by broadcasting to all the receivers the anchor element of the chain, for example s_0, signed with its private key (in case of public-key based bootstrapping), or by encrypting s_0 with the secret key it shares with each receiver in the network (for symmetric-key based bootstrapping).

For each packet generated in time slot i, the source uses the authentication key K_i to compute a MAC on the packet. The MAC is then appended to the packet, which is transmitted to the receiver(s). When a node receives a packet, it first checks whether the packet is *fresh*, that is, it was sent in a time interval whose corresponding TESLA key has not been disclosed. This is the fundamental security criterion in TESLA. Each receiver discards any packet that does not meet the security criterion, and buffers only the packets that satisfy the freshness condition. The receiver cannot authenticate the packets immediately since it does not know the corresponding key K_i. The sender discloses the key K_i at a later instant in time by broadcasting the corresponding key seed s_i. Upon receiving s_i, each receiver first verifies the authenticity of s_i by checking $s_i \xrightarrow{F_1} s_{i-1}$ (and therefore ultimately verifying against the anchor element s_0 which has already been authenticated). If s_i verifies correctly, each receiver can compute K_i: $s_i \xrightarrow{F_2} K_i$ and subsequently use the computed K_i to verify the MAC on the packets received during interval i.

Once s_i is disclosed, any node with knowledge of s_i can compute K_i and attempt to masquerade as the sender by forging MACs using K_i. Therefore, K_i is used to compute MACs on packets generated only during the interval i, other time intervals use different keys to compute the MACs. The key seed s_i is disclosed only d time slots after i so that no malicious node can compute K_i and forge packets in the intervening period. d is computed based on the maximum network delay from the source to all the receivers. This is the principle of delayed disclosure of keys.

The major advantage of TESLA in this regard is that it allows similar authentication through the use of computationally efficient MAC functions, and is therefore very attractive for authentication in devices of limited capabilities.

The above is a basic description of TESLA. The algorithm has several enhancements to mitigate various drawbacks, they are described in [16].

III. Review of TESLA Certificate and its Application to a Hybrid Network Topology

The idea of certificates based on TESLA was proposed in [14]. The idea has been formalized to form a TESLA-based PKI in [13]. In the algorithm described in [13], there is a certificate authority CA who creates certificates for an entity B. A low-powered device D contacts B to use its service. The CA and B initially share a secret key $K_{CA,B}$. During time slot n, the CA generates authentication key aK_{B_n} for B to use to compute the MAC on its messages in that interval. The CA creates a certificate $Cert_{CA_n}(B)$ to bind aK_{B_n} to B for interval n. The CA uses its TESLA key tK_{CA_n} to encrypt aK_{B_n} in the certificate, and uses the same key to compute a MAC on the different fields in the certificate.

$$Cert_{CA_n}(B) = \left(ID_B, \{aK_{B_n}\}_{tK_{CA_n}}, n+d, MAC_{tK_{CA_n}}(..)\right) \quad (1)$$

Equation 1 represents the TESLA certificate for node B. aK_{B_n} is known only to the CA and B during period n, while tK_{CA_n} is known only to the CA. $n+d$ indicates the time at which the CA will disclose tK_{CA_n} to the nodes, that is, it is the expiration time of the certificate. The CA sends $Cert_{CA_n}(B)$ to B alongwith aK_{B_n}, which is encrypted with $K_{CA,B}$.

In the time interval $\langle n, n+d\rangle$, D sends a request to B for using B's service:

$$D \rightarrow B\colon\ (request) \quad (2)$$

To authenticate itself to D, B sends an authentication packet containing its certificate and a MAC on the request, computed with aK_{B_n}.

$$B \rightarrow D\colon\ \left(Cert_{CA_n}(B), MAC_{aK_{B_n}}(request)\right) \quad (3)$$

When D receives the authentication message, it checks the timestamp of $Cert_{CA_n}(B)$ to make sure it has arrived before time $n+d$, when the CA discloses tK_{CA_n}. If the certificate is "fresh", D buffers the authentication packet. At time $n+D$, the CA discloses its TESLA key tK_{CA_n}. Upon receiving the key, D verifies $Cert_{CA_n}(B)$ by checking the MAC in the certificate using tK_{CA_n}. If the MAC verifies correctly, D obtains B's authentication key aK_{B_n} from the certificate by decrypting with tK_{CA_n}. Subsequently, D checks $MAC_{aK_{B_n}}(request)$ to verify the authenticity of B. Therefore, D is able to verify the identity of B only if it receives $Cert_{CA_n}(B)$ before $n+d$. Once the CA discloses its TESLA key tK_{CA_n}, any node could forge a certificate for the time interval n.

The TESLA certificate algorithm described above has several shortcomings. The authors propose that the CA should generate the subject B's authentication key aK_{B_n}, and subsequently send it to B securely, alongwith the corresponding TESLA certificate. Therefore the CA could easily fake messages as coming from B. In traditional public-key cryptography, the CA does not get to know the private key of any party. Node B will only send the corresponding public key to the CA for certificate generation. Also, the approach of CA generating the authentication key of B makes it necessary to bootstrap the CA and B with a shared secret $K_{CA,B}$. In a network with a large number of nodes, this bootstrap operation should ideally be avoided.

A TESLA certificate allows a node B to add authentication to packets for a single period in time. As the authors mention in [13], the lifetime of the certificate is short. Therefore, a source node B that transmits for multiple time intervals will need several TESLA certificates from the CA. If there are many sources that send data over long intervals, this can add up to a substantial overhead.

The authors in [13] propose an algorithm for mobile node handoff when node D moves from the range of access point B to the range of access point B'. To authenticate D with node B', the CA gives node D a certificate $Cert_{CA}(D)$:

$$Cert_{CA}(D) = (ID_D, \{aK_{AP,D}\}_{gK_{AP}}, TS_A, SIGN_{-K_{CA}}(..)) \quad (4)$$

$Cert_{CA}(D)$ is used by D to establish a shared key $K_{B',D}$ with the new access point, as described in algorithm 1 in [13]. This algorithm is not secure. The key gK_{AP} is known to all access points. Therefore any access point B with whom D has communicated in the past, or who overhears the communication between D and another access point, can derive $aK_{AP,D}$ from $Cert_{CA}(D)$. Consequently, any access point B who eavesdrops on the handoff algorithm between D and access point B', can obtain the new key $K_{B',D}$ from the message in step 3 of algorithm 1 [13].

The authors describe an application of TESLA certificates for authentication in hierarchical ad hoc sensor networks in [15]. The focus of the work is on authentication between sensor nodes and the base stations/applications, that is, point-to-point authentication between nodes of varying capabilities. The paper does not address authentication between peer nodes, or authentication in group communication.

The correct operation of TESLA certificates depends on the secure delayed disclosure of signing key tK_{CA_n} by the CA. Neither paper mentions how the key disclosure message from the CA will be trusted by the nodes - a mechanism for bootstrapping the messages from the CA is required, but it is not explicitly mentioned. It would be possible, given the assumptions and constraints in the papers, that the CA authenticates the key disclosure message with the secrets it shares with every node in the network, but this approach is not scalable. The rest of this section discusses security issues concerning the algorithms presented in [15].

The authentication framework in [15] requires a trusted third party (TTP) to generate the initial certificate $iCert_{TTP}(.)$ for every node. A node D needs a different $iCert_{TTP}(D)$ for every AP it communicates with; unless D knows in advance all the APs it will talk to in the network and therefore obtains corresponding $iCert_{TTP}(D)$s from the TTP, it needs to contact the TTP every time it moves to a new AP. The framework has different entities generating the certificates, either public key-based or TESLA-based, for different nodes - there are hence multiple CAs at different levels in the network. This leads to a complex setup at best, and incompatible security policies at worst. We believe this structure can be simplified greatly, with only one CA. Also, the paper mentions that the application A generates the key $K_{B,D}$ to be used between a base station B and sensor node D. While this is acceptable if B and D trusts A completely, it is preferable that the key be generated by the parties that will use it, if they have the resources to do so without additional overhead (B can certainly generate the key easily).

We found problems with the data origin authentication algorithms - both the *weak* mode and the *assured* mode. Since the source and the destination are D and A respectively, it is sufficient to include $MAC_{K_{A,D}}(..)$ only in the message - addition of $MAC_{K_{B,D}}(..)$ adds overhead without adding security. Also, the reason for allowing base station B to add $MAC_{K_{A,B}}(..)$ is unclear - B is only a forwarding node in this scenario and does not generate or modify the message in any way. It is not necessary for verification at A that B has checked the message - it would suffice for B to drop the message if it could not verify (or allow A to do all the verification itself). This can lead to a denial of service (DoS) attack at A from a malicious B, which can send a large number of intercepted or spoofed messages with incorrect MACs. Also, even if the message from D is correct and has not been modified at all, A can reject the message if the $MAC_{K_{A,B}}(..)$ is incorrect for some reason.

In the *assured* mode of data delivery, the shared key $K_{C,D}$ is not needed if the forwarding node (FN) C uses TESLA certificates also, which is desirable. However, the authors make the strong assumption that the FNs have enough capabilities to support public key cryptography. Encrypting the random nonce r_n using both $K_{C,D}$ and $K_{B,D}$ is wasteful. Instead, it would be better to have two random nonces r_n and r'_n, separately encrypted with $K_{B,D}$ and $K_{C,D}$, so that D can be assured the data was correctly received by both FN and AP when it gets back the nonce responses from both. In either mode, A does not send any information to D on successful acceptance of the message from D, a message is sent on reject only. Even in the reject message, the reason for including $MAC_{K_{A,B}}(dRej, B)$ is unclear, and the addition of $MAC_{K_{B,D}}(dRej, B)$ is superfluous.

An important point to note here is that TESLA certificates do not provide non-repudiation; once the keys are disclosed, anyone can forge MACs on spoofed messages and therefore the authenticity of messages created in the past cannot be guaranteed. The authors also mention that revocation of certificates is another outstanding issue.

IV. Modifications to the Original TESLA Certificate Approach

We propose several modifications to the algorithms in [13], [15] to make the authentication framework using TESLA certificates more secure. In the following description, D is a sensor node, B is an access point (AP) and A is an application. We assume that all nodes in the network, including sensors and access points, know the public key $+K_{CA}$ of the CA.

A. Creation of TESLA Certificate $Cert_{CA_n}(D)$ for Node D

The authentication key aK_{D_n} that D uses to compute the MACs in time period n, is generated by D itself using a one-way function F, applied to a secret seed s_l:

$$s_l \xrightarrow{F} aK_{D_n} \quad (5)$$

D applies a collision-resistant hash function H to aK_{D_n} and generates the hash aK'_{D_n}:

$$aK_{D_n} \xrightarrow{H} aK'_{D_n} \quad (6)$$

We assume that F and H are publicly available.

Subsequently D sends aK'_{D_n} to the CA for certificate generation. This message can be broadcast in the clear, since knowledge of aK'_{D_n} does not allow any other node to generate aK_{D_n}, owing to the one-way property and collision-resistance of H. The CA generates the TESLA certificate for node D for period n using aK'_{D_n} and sends it to D:

$$Cert_{CA_n}(D) = \left(ID_D, \{aK'_{D_n}\}_{tK_{CA_n}}, n+d, MAC_{tK_{CA_n}}(..)\right) \quad (7)$$

$$CA \rightarrow D : \left(Cert_{CA_n}(D), SIGN_{-K_{CA}}(..)\right) \quad (8)$$

D can check the authenticity of the TESLA certificate by verifying the signature attached to the message using the CA's public key $+K_{CA}$. Although digital signature verification might be computationally expensive for D, this is done only when it gets new TESLA certificates from the CA. (In section V we extend the TESLA certificate so that this signature verification by D needs to be done rarely.) Subsequently D can use $Cert_{CA_n}(D)$ for authenticated communication with other nodes in the network. When sending message m, D uses the key aK_{D_n} to compute $MAC_{aK_{D_n}}(m)$. A receiving node R buffers m until it can be authenticated. When the CA discloses tK_{CA_n}, D sends aK_{D_n} to R. R can obtain aK'_{D_n} from the certificate using tK_{CA_n} and verify that $aK_{D_n} \xrightarrow{H} aK'_{D_n}$. If verification succeeds, R computes $MAC_{aK_{D_n}}(m)$ and accepts m if the MAC is correct.

The important assumption here is that R has some mechanism for verifying the authenticity of tK_{CA_n} (that is, tK_{CA_n} has been disclosed by the CA and not by some other entity). One can achieve this if the CA discloses tK_{CA_n} by encrypting it with the shared secret of every node. The alternative is that the CA sign the disclosure message using $-K_{CA}$. Since every node knows $+K_{CA}$, the message can be easily verified. This adds computational burden at the nodes for signature verification. In section V we discuss mechanisms on how this computation expense can be amortized.

In the above algorithm, the CA does not get to know the authentication key aK_{D_n}, as we have intended. Also, the nodes do not need to share secrets with the CA to authenticate messages from the CA, or to send it messages. This comes at the expense of one additional key disclosure by the source, and the computation overhead at nodes to verify CA signatures.

B. Secure Mobile Node Handoff

The second improvement we suggest is to make the mobile node handoff algorithm in [13], [15] more secure. Here the requirement for a symmetric shared key between D and the CA is important, and accordingly we assume that D has established a shared key $K_{CA,D}$ with the CA (either offline, or during bootstrapping). Also, an access point B has a public-private key pair $\langle +K_B, -K_B \rangle$. When node D moves from access point B to access point B', it contacts the CA requesting a certificate to authenticate itself to B'. The CA generates $Cert_{B'}(D)$ so that D can authenticate itself to B' and sends $Cert_{B'}(D)$ to D, alongwith the key $aK_{AP_{B'D}}$ to be used for secure communication between B' and D.

$$CA : Cert_{B'}(D) = \left(ID_D, \{aK_{AP_{B'D}}\}_{+K_{B'}}, TS_A, SIGN_{-K_{CA}}(..)\right) \quad (9)$$

$$CA : K_{CA,D} \xrightarrow{F} K'_{CA,D} \quad (10)$$

$$CA \rightarrow D : \left(Cert_{B'}(D), \{aK_{AP_{B'D}}\}_{K_{CA,D}}, MAC_{K'_{CA,D}}(..)\right) \quad (11)$$

Subsequently, D communicates with B' using the mobile node handoff algorithm, with the certificates described above. In step 3 of the algorithm in [13], [15], B' sends the following message to D:

$$B' \rightarrow D : \left(hoOK, \{K_{B'D}\}_{aK_{AP_{B'D}}}, MAC_{aK_{B'_n}}(..)\right) \quad (12)$$

The above modifications ensure that the key $K_{B'D}$ is known only to B' and D, and not to any other access point B.

Several other modifications are possible to the algorithms mentioned in [15], following the ideas of the modifications described above. They are mostly straightforward, and we omit a description of them for brevity.

V. Extending TESLA Certificates

We consider a network scenario where a group of wireless mobile nodes in the network take part in communication with one another. The wireless mobile nodes are resource-constrained devices of limited computational power, storage and finite energy (for example, battery source with no re-charging capability). The mobile nodes do not have any pre-existing security information about one another. However, all the nodes have access to an online certificate authority, and the public key $+K_{CA}$ of the CA is available to every node. We assume that all the nodes are loosely time-synchronized with the CA. The CA can communicate with the entire network simultaneously through wireless broadcast channels. The wireless transmission channels are assumed to be error-free, so that control messages or data packets do not get lost. We also assume that appropriate policies are in place to allow each node to securely identify itself to the CA during the initial bootstrapping phase, and each node A shares a unique secret key $K_{CA,A}$ with the CA.

Our objective is to design an authentication mechanism that allows any receiver in the network to securely authenticate messages from a sender node with limited expenditure of processing power and energy. The receiver should also be able to authenticate buffered messages from a source with which it might not be in communication contact at the time of message authentication (for example, due to network partition, etc.). The receiver does not need to trust the source or have any prior information about the source; the only requirement, as stated above, is that the receiver trust the CA (or at least an entity who can prove knowledge of $-K_{CA}$, the private key corresponding to $+K_{CA}$). We assume that one-way functions F_1, F_2 and F_3, derived from pseudo-random function (PRF) families, are publicly available.

A. Authentication for Unicast Communication

1) Bootstrapping of the Source Node and the Certificate Authority: We make use of the TESLA key chain generation described in [14], [16]. Initially, the source node A generates a random seed $s_{A,n}$ and applies one-way function F_1 to $s_{A,n}$ to form a *hash chain*:

$$s_{A,0} \xleftarrow{F_1} s_{A,1} \xleftarrow{F_1} ... \xleftarrow{F_1} s_{A,n-1} \xleftarrow{F_1} s_{A,n} \tag{13}$$

The value n depends on the number of time intervals in which A expects to be a source. If the duration of each time interval is Δ, and the total time of A's transmission is T, we have $n = \frac{T}{\Delta}$. A subsequently applies F_2 to each key $s_{A,i}$ generated above and obtains the output $s'_{A,i}$.

$$\begin{array}{ccccccccc} s_{A,0} & \xleftarrow{F_1} & s_{A,1} & \xleftarrow{F_1} & ... & \xleftarrow{F_1} & s_{A,n-1} & \xleftarrow{F_1} & s_{A,n} \\ \downarrow F_2 & & \downarrow F_2 & & ... & & \downarrow F_2 & & \downarrow F_2 \\ s'_{A,0} & & s'_{A,1} & & ... & & s'_{A,n-1} & & s'_{A,n} \end{array} \tag{14}$$

A applies F_3 to $s'_{A,0}$ to obtain $h_{A,0}$:

$$s'_{A,0} \xrightarrow{F_3} h_{A,0} \tag{15}$$

In time period t_0, A sends $h_{A,0}$ to the CA for obtaining a TESLA certificate. On successful verification of A's identity, the CA generates a TESLA certificate for A:

$$Cert_{CA}(A) = \left(ID_A, \{h_{A,0}\}_{tK_{CA_0}}, t_0 + d, MAC_{tK_{CA_0}}(..)\right) \tag{16}$$

$$CA \rightarrow A : \left(Cert_{CA}(A), SIGN_{-K_{CA}}(..)\right) \tag{17}$$

Here d is the key disclosure delay for the CA TESLA signature key, and $tK_{CA,0}$ is the CA MAC key for the time period $\langle t_0, t_0 + d\rangle$.

$tK_{CA,0}$ is generated by the CA using the one-way chain algorithm. The CA starts with an initial seed $s_{CA,n}$ and generates $tK_{CA,0}$ as follows:

$$\begin{array}{ccccccc} s_{CA,0} & \xleftarrow{F_1} & ... & \xleftarrow{F_1} & s_{CA,n-1} & \xleftarrow{F_1} & s_{CA,n} \\ \downarrow F_2 & & ... & & \downarrow F_2 & & \downarrow F_2 \\ tK_{CA,0} & & ... & & tK_{CA,n-1} & & tK_{CA,n} \end{array} \tag{18}$$

2) Message Transmission from Source to Receiver: Let A send messages to receiver node B starting in the time interval $\langle t_0, t_0 + d\rangle$. A computes a MAC over the message m_0 using $s'_{A,0}$ and includes its TESLA certificate $Cert_{CA}(A)$ with the message it sends to B:

$$A \rightarrow B\colon \{M_0 | M_0\colon \left(m_0, MAC_{s'_{A,0}}(m_0), Cert_{CA}(A)\right)\} \tag{19}$$

B checks the freshness of the certificate by checking the timestamp of $Cert_{CA}(A)$ to make sure it has arrived before time $t_0 + d$. If $Cert_{CA}(A)$ has arrived within $\langle t_0, t_0 + d\rangle$, B stores M_0 in its buffer, else B discards the message.

Checking the timestamp on $Cert_{CA}(A)$ is critical for the security of our algorithm. Once the CA discloses $s_{CA,n-1}$ at time $t_0 + d$, any node in the network can create a fake certificate with timestamp $t_0 + d$, allegedly generated by the CA. Therefore receivers will only accept certificates for which the CA TESLA key has not been disclosed at the time of receiving the certificate.

3) Message Authentication at Receiver: At time $t_0 + d$, the CA broadcasts the key $tK_{CA,0}$ to the network:

$$CA \rightarrow network\colon \left(\langle t_0, t_0 + d\rangle, s_{CA,0}, SIGN_{-K_{CA}}(..)\right) \tag{20}$$

Node A also receives the CA broadcast. Subsequently, it transmits $s_{A,0}$ to node B. Receiver B checks the authenticity of the CA broadcast by verifying the signature using $+K_{CA}$. If verification is successful, B checks the MAC on $Cert_{CA}(A)$ using $tK_{CA,0}$, which is derived from $s_{CA,0}$ that is obtained from (20). If the MAC is correct, B obtains $h_{A,0}$ from $Cert_{CA}(A)$ by decrypting with $tK_{CA,0}$. B then checks if the key $s_{A,0}$ received from A is correct:

$$\begin{aligned} s_{A,0} &\xrightarrow{F_2}\colon s'_{A,0} \\ s'_{A,0} &\xrightarrow{F_3}\colon h'_{A,0} \\ h'_{A,0} &\stackrel{?}{=} h_{A,0} \end{aligned} \tag{21}$$

If the above check returns a positive result, then B checks $MAC_{s'_{A,0}}(m_0)$ using $s'_{A,0}$ and accepts m_0 if the MAC verifies correctly. B also stores in memory the CA key broadcast message (and therefore $s_{CA,0}$), $Cert_{CA}(A)$ and the initial key $s_{A,0}$ of A's hash chain. Figure 2 gives a timing diagram representation of the protocol steps till the initial packet authentication at the receiver.

Messages from A to B in subsequent time intervals use the corresponding key of A's key chain to compute the MAC. A does not have to include its TESLA certificate in messages subsequent to M_0, under the assumption that every receiver has received M_0 correctly. For example, in the period $\langle t_i, t_i + \Delta\rangle$, message M_i from A to B would look like:

$$A \rightarrow B\colon \{M_i | M_i\colon \left(m_i, MAC_{s'_{A,i}}(m_i)\right)\} \tag{22}$$

At time $t_i + d$, A transmits $s_{A,i}$ to B. B can check the correctness of $s_{A,i}$ immediately by verifying $s_{A,i} \xrightarrow{F_1} s_{A,i-1} \xrightarrow{F_1} ... \xrightarrow{F_1} s_{A,0}$. Since $s_{A,0}$ has already been verified, and F_1 is a secure one-way function, the above check will verify that $s_{A,i}$ belongs to A's key chain. However, if B wants to be additionally careful, it can verify $s_{A,i}$ going through all the steps outlined above, using the CA key broadcast message and $Cert_{CA}(A)$.

It is to be noted that A has to wait at least time $t_0 + d$ before it can disclose its initial TESLA key, since the CA sends its TESLA key $s_{CA,0}$ at time $t_0 + d$. For subsequent

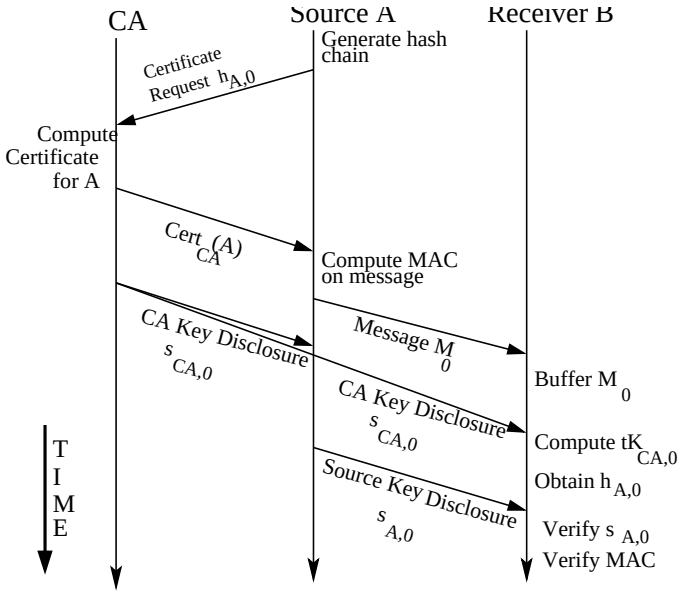

Fig. 2. Authentication of packets using extended TESLA certificate: time diagram

key disclosures by A, it does not have to wait d time units to disclose its TESLA key for the corresponding interval. However, we maintain the delay to allow a consistent rate for key disclosure, and for verification at the receiver.

Thus messages from A to B can be authenticated. The above algorithm requires A to perform one signature verification, and B also has to perform one signature verification on the initial key disclosure message from the CA. All other messages can be authenticated using low-computation symmetric MACs. A and B also does not need to perform clock synchronization directly with one another, thereby saving on additional message rounds and protocol complexity (and possibly also on the cyclical dependency between authentication and clock synchronization).

The CA need not be on-line all the time and does not need to broadcast frequent key disclosure messages. However, if the security policy demands so, the CA can periodically generate new TESLA certificates for a source, and broadcast periodic key disclosure messages. After the initial key disclosure message from the CA signed with $-K_{CA}$, subsequent key disclosure messages from the CA can be authenticated using one-way chains. For example, CA discloses the key $s_{CA,i}$ in period $\langle t_i, t_i + d\rangle$. Receiver B can verify that $s_{CA,i}$ belongs to CA's one-way chain: $s_{CA,i} \xrightarrow{F_1} s_{CA,i-1} \xrightarrow{F_1} ... \xrightarrow{F_1} s_{CA,0}$, where $s_{CA,0}$ has been verified before using $+K_{CA}$. B does not need to check CA's signature to verify $s_{CA,i}$.

4) Revocation of TESLA Certificates: The CA might need to broadcast a certificate revocation message at any time circumstances warrant that the TESLA certificate of a node has to be revoked. Assume the CA revokes the TESLA certificate of node A in the time period $\langle t_i, t_i + d\rangle$. Then the CA broadcasts the following message to the network:

$$CA \rightarrow network: \\ (\langle t_i, t_i + d\rangle, REVOKE\left(Cert_{CA}\left(A\right)\right), s_{CA,i}, \\ MAC_{s_{CA,i+1}}\left(..\right)) \quad (23)$$

The receiver buffers the message and waits for the CA to disclose $s_{CA,i+1}$. The traffic received from A in the intermediate period is also buffered, awaiting the verification of the revocation message, due to the possibility that the revocation message might be a fake.

The CA discloses $s_{CA,i+1}$ with the next message it broadcasts to the network. The receiver can verify the authenticity of $s_{CA,i+1}$ and therefore the revocation message by verifying the correctness of the one-way chain:

$$s_{CA,i+1} \xrightarrow{F_1} s_{CA,i} \xrightarrow{F_1} .. \xrightarrow{F_1} s_{CA,0} \quad (24)$$

where $s_{CA,0}$ has been verified using $+K_{CA}$ from the initial key disclosure broadcast message of the CA. If the revocation message is correctly verified, the receiver discards the buffered messages from A and adds the sender to the revoked users list.

The revocation message can be merged with the key disclosure message, the combined message can look like:

$$CA \rightarrow network: \\ (\langle t_i, t_i + d\rangle, REVOKE(..), s_{CA,i}, \\ MAC_{s_{CA,i+1}}\left(..\right), SIGN_{-K_{CA}}\left(..\right)) \quad (25)$$

where the REVOKE field will contain the TESLA certificates to be revoked, the MAC is computed on the revoked certificates and the signature verifies $s_{CA,i}$ for nodes that might need the verification.

Non-repudiation is not provided by the authentication algorithm we have described in this section. We are currently investigating efficient additions to the algorithm to provide this important feature.

5) Dealing with Network Partition or Receiver Connectivity Failure: Due to the conditions of the wireless medium and node mobility, it might happen that the source and the destination lose the communication path between them. Assume a scenario where the receiver B has buffered packets from the source A in the time interval $\langle t_i, t_i + \Delta\rangle$, but has lost communication with the source at time $t < t_i + d$. Also, B had initially successfully received the key disclosure message from the CA and had successfully verified packets from A transmitted in time $t < t_i$. A broadcasts the MAC key $s_{A,i}$ of period $\langle t_i, t_i + \Delta\rangle$ at time $t_i + d + \delta$, where δ is a small increment in time. Since the message is a wireless broadcast, nodes which can communicate with the source at time $t_i + d + \delta$ receive the broadcast. These nodes cache the authentication key for a specific time period. Node B does not receive the message from A; therefore it broadcasts a request to its neighbors at time $t_i + d + \delta$ for A's authentication key for time-period $\langle t_i, t_i + \Delta\rangle$. The request is propagated by B's neighboring nodes throughout the network till it reaches one or

more nodes that have cached $s_{A,i}$. These nodes send a reply to B with the key $s_{A,i}$. B can subsequently verify the messages received from A during $\langle t_i, t_i + \Delta \rangle$. If no reply is received, B keeps the messages in the buffer in the expectation of receiving a reply in the future, or re-establishing communication with A at a later time. Any authentication key $s_{A,k}(k > i)$, received either directly from A or through other nodes at any time $t > t_i + d$ can authenticate messages received in $\langle t_i, t_i + \Delta \rangle$, provided $s_{A,i}$ and $s_{A,k}$ belong to the same one-way chain. If the buffered messages cannot be verified, B can erase them when space constraints in the buffer arise.

The above request-reply message exchange has the additional benefit that it can help to re-establish communication between A and B through new paths.

A different scenario might be that the receiver B has gone offline temporarily after receiving the initial few packets from the source A, but before the initial key disclosure message from the CA. Assume B returns online at time $t > t_0 + d$. B has buffered packets from A for period $\langle t_0, t_0 + \Delta \rangle$, but cannot verify them since it has missed both the key disclosure message from the CA and the $s_{A,0}$ key broadcast message from A. B broadcasts a request for the CA message and A's key to its neighbors; as in the previous discussion, B gets a reply back from any node that has cached the messages. B is subsequently able to authenticate the packets from A.

B. Group-based Certificates for Authentication in Multicast Communication

The algorithm described in section V-A can authenticate messages from a source to a multicast group comprised of nodes in the wireless network. The source obtains TESLA certificate from the CA as described in V-A, and multicasts data packets to the group members. The group members buffer the data packets till they receive the authentication keys. When the CA broadcasts the key disclosure message, the source multicasts the MAC computation key, and the receivers can authenticate the buffered packets as in section V-A.

In a manner similar to above, multiple sources send authenticated messages to the group. The group members do not need to exchange security information with one another apriori, but they need to be able to receive messages from the CA, and know the CA's public key $+K_{CA}$. In this approach, the CA has to generate as many certificates as there are sources for a group. If there are p simultaneous groups, each with q sources on average, there will be pq certificates in the network on average. To have fewer certificates in the network, we propose the use of *group-based certificates*, described in the following sections.

1) Multiple Sources in One Multicast Group: We illustrate the concept of group-based certificates by initially considering one multicast group with multiple senders in the network. In the group-based certificate approach, the CA creates certificates based on two fields - the sources for a given group, and the group identifier. For example, say group G has three sources A, B and C, each of which sends the authentication bootstrapping key $h_{A,0}, h_{B,0}$ and $h_{C,0}$ respectively to the CA. The CA creates the group certificate $GCert_{CA}(G)$ for group G and broadcasts $GCert_{CA}(G)$ to the network.

$$GCert_{CA}(G) = \big(G, t_0 + d, \langle ID_A, \{h_{A,0}\}_{tK_{CA,0}} \rangle, \langle ID_B, \{h_{B,0}\}_{tK_{CA,0}} \rangle, \langle ID_C, \{h_{C,0}\}_{tK_{CA,0}} \rangle, MAC_{tK_{CA,0}}(..)\big) \quad (26)$$

$$CA \rightarrow network : (GCert_{CA}(G), SIGN_{-K_{CA}}(..)) \quad (27)$$

Since $GCert_{CA}(G)$ is not specific to any particular source, there is no need to send it to the multicast source nodes. Broadcast from the CA to the network has the advantage that all the network nodes who are subscribed to the group, either as receivers or as sources, will receive the certificate; it also removes the burden from the sources to make sure their certificates reach all the group receivers, and has the added advantage of making the source messages smaller in size.

The multicast group members can verify that $GCert_{CA}(G)$ is generated by the CA by performing signature verification on the CA broadcast in (27) using $+K_{CA}$. When a source sends multicast data packets to the group members in G: $* \rightarrow G$: $\{M_0 | M_0 : \Big(m_0, MAC_{s'_{*,0}}(m_0)\Big)\}$ (where $*$ stands for any source in G), the receivers buffer the data packets till the CA discloses the key $s_{CA,0}$. A source A then broadcasts the authentication key $s_{A,0}$ to the group. Receivers can verify the buffered data packets using $s_{CA,0}$ and $s_{A,0}$ as described in previous sections.

The group-based certificates can be used to perform source access control to a group at the granularity of time period d. If the source set for a group G changes dynamically, the CA can generate and broadcast $GCerts$ every d time units, with each $GCert$ containing the list of sources valid for the subsequent time period. Newly-added sources will have their data packets buffered till the CA key disclosure message, and the following authentication key broadcast by the new sources. Sources who have been in previous $GCerts$ from the CA can have their packets immediately authenticated using the hash chain. Any source which is not included in the most recent $GCert$ will have its packets discarded by the receivers.

2) One Node as Source in Multiple Groups: A node might be a source in more than one multicast group. It might send traffic to different groups at different rates and instants in time. The node should not use the same hash chain to authenticate its messages to the different groups, since it might leave it vulnerable to attacks from malicious nodes that act as receivers and sources in the same set of multicast groups. For example, assume node A is a source for two groups $G1$ and $G2$. It sends data to $G1$ at a faster rate compared to $G2$. Assume A uses the same hash chain to authenticate messages to both the groups. A malicious node X is a receiver in group $G1$, and a source in group $G2$. X will receive elements of A's hash chain when A broadcasts the elements to receivers in $G1$. Since the transmission rate to $G2$ is slower, depending on

the difference in rate, the same hash chain elements might be valid authentication keys in $G2$ (that is, not yet disclosed to members in $G2$). X can therefore use the hash chain elements it obtained in $G1$ to send spurious messages to $G2$, and the messages will be authenticated as coming from A. This security vulnerability can be avoided if the CA and/or the source maintains a uniform rate at which keys are used and disclosed across all the groups. However, maintaining uniform rate would disadvantage groups where the transmission is faster compared to other groups. A preferable solution would be for the source A to have different hash chains for the different groups. A can start with the same seed $s_{A,x}$ and generate a different chain for every group G_i ($i \in \{1..L\}$ where L is the total number of groups) by concatenating $s_{A,x}$ with G_i and applying F_1 to the result of the concatenation:

$$
\begin{array}{ccccccc}
s_{G_iA,0} & \xleftarrow{F_1} & \dots & \xleftarrow{F_1} & s_{G_iA,n} & \xleftarrow{F_1} & s_{A,x}||G_i \\
\downarrow F_2 & & \dots & & \downarrow F_2 & & \\
s'_{G_iA,0} & & \dots & & s'_{G_iA,n} & & \\
\dots & & \dots & & \dots & & \\
s_{G_jA,0} & \xleftarrow{F_1} & \dots & \xleftarrow{F_1} & s_{G_jA,n} & \xleftarrow{F_1} & s_{A,x}||G_j \\
\downarrow F_2 & & \dots & & \downarrow F_2 & & \\
s'_{G_jA,0} & & \dots & & s'_{G_jA,n} & &
\end{array}
$$

$$
\begin{array}{l}
s'_{G_iA,0} \xrightarrow{F_3} h_{G_iA,0} \\
s'_{G_jA,0} \xrightarrow{F_3} h_{G_jA,0}
\end{array}
\tag{28}
$$

where $i, j \in \{1..L\}$.

A sends the hash elements $h_{G_iA,0}$ and $h_{G_jA,0}$ to the CA to be included in $GCert_{CA}(G_i), GCert_{CA}(G_j)$ respectively. Subsequently, the messages A sends to groups G_i and G_j are authenticated using elements from the hash chain corresponding to G_i, G_j respectively.

The concept of generating multiple hash chains from the same seed is similar to the *concurrent TESLA instances* described in [16], where the authors use different TESLA chains based on the transmission rates for different receivers. However, the authors limit the concept to one group and do not extend to the case of multiple groups as above.

VI. Security Analysis of Extended TESLA Certificate Approach

The modified TESLA certificate approach provides strong authentication guarantees and is resistant to active attacks by malicious nodes in the network. In the following, we highlight the security features of the modified TESLA certificates, for both unicast and multicast communication described in section V. We assume that the CA is always secure, since compromise of the CA is a single point of failure in the network, and will nullify most security properties of our algorithm.

A. Malicious Node with Connectivity to Source and Receiver

We consider the case where a malicious node in the network attempts to create fake packets from a source to the receiver(s). We assume that the malicious node X can hear packet transmissions from the actual source A, and can also transmit to the receiver B. X can also receive the broadcast messages from the CA. Therefore, shortly after time $t_0 + d$, X has knowledge of $Cert_{CA}(A)$, message M_0 from A to B, $s_{CA,0}$ broadcast by the CA, and $s_{A,0}$ sent by A. X can verify that $s'_{A,0}$ belongs to the authentication hash chain of A by performing the verification procedure of equation (21). Having obtained a verified element of A's authentication chain, X can attempt to spoof messages as coming from A, staring at time $t_0 + k\Delta$, where $\frac{d}{\Delta} = k$. To achieve this, X needs to generate $s_{A,k}$ from $s_{A,0}$ where $s_{A,k} = F_1^{-k}\{s_{A,0}\}$. Due to the one-way property of F_1, this is computationally infeasible for X, and is of complexity $O\left(2^K\right)$, where each element of the hash chain is K bits and K is assumed to be large. Without a valid $s_{A,k}$, it would be impossible for X to spoof a message that would be successfully authenticated by B.

X could also attempt to spoof packets from A at any time between $\langle t_0, t_0 + d\rangle$. This would require that X successfully generate an element of A's hash chain without knowledge of any legitimate element of the hash chain. This has the same computational complexity of $O\left(2^K\right)$ and is computationally infeasible for any X with finite resources.

A third approach X could attempt would be to generate an independent hash chain that produces the hash value $h_{X,0}$ that is computationally indistinguishable from $h_{A,0}$. This would allow X to use element $s_{X,0}$ of its own hash chain to authenticate messages purportedly generated by A. However, this is computationally infeasible due to the collision-resistance and strong one-way property of F_1, F_2 and F_3.

Failing any attack on A's hash chain as above, X could attempt to masquerade as the CA and generate a fake certificate for A as in equation (16), and also generate fake CA key disclosure broadcast message similar to equation (20). However, unless X knows the CA private key $-K_{CA}$, it will not be able to correctly sign the fake $Cert_{CA}(A)$, and therefore the fake certificate will be rejected by A. Likewise, the fake CA broadcast message from X will be rejected by the receivers unless the signature in the message is verified as correct using $+K_{CA}$. As per our assumption of the security of the CA, $-K_{CA}$ is known only to the correct CA, and therefore X would not be successful in this attack.

X could attempt to fake CA key disclosure messages subsequent to (20), but (a) the fake hash element $s_{CA_X,i}$ will not verify successfully to the anchor element $s_{CA,0}$ and (b) this still does not allow X to fake elements of A's hash chain. At best, X would be able to confuse the receivers temporarily and therefore launch a DoS attack for a limited time.

Once the CA has disclosed its TESLA key $s_{CA,i}$ for period $\langle t_i, t_i + d\rangle$, any node in the network can create a fake certificate, purportedly generated by the CA before $t_i + d$, by computing $tK_{CA,i}$ from $s_{CA,i}$. However, when the fake

certificate is sent to any receiver, it will arrive after time $t_i + d$, and therefore be rejected, as per the security requirement in section V-A.2.

B. Attack on the CA Revocation Messages

A malicious node X in the network can attempt to broadcast fake revocation messages, similar to (23), and thereby attempt to disqualify legitimate sources in the network. To generate a fake revocation message that will be successfully accepted by the receivers, X should be able to compute a MAC on the fake revocation message using the key $s_{CA,i+1}$, with knowledge of at most the key $s_{CA,i}$. Using reasoning similar to the previous section, owing to the one-way property of F_1, this has computational complexity $O\left(2^K\right)$ and is infeasible for X. At most, X can trick the receivers in buffering the fake revocation message, till the next message disclosure from the CA, when the MAC on the fake message will not verify correctly using the recently disclosed (correct) $s_{CA,i+1}$, and therefore be discarded.

C. Attack on Group Certificates

Group certificates are based on the construction of the single-source certificates. Due to the security of the single-source certificates discussed in the previous sections, it can be shown with minor additions that the group-based certificate approach guarantees the security of source and message authentication in a multicast group setting, for both cases of multiple sources in a group, and one source in multiple groups.

Based on the security analysis above, the extended TESLA certificate approach is secure against message spoofing attacks by malicious nodes in the network.

VII. Comments on the Extended TESLA Certificate Approach

A. Performance Analysis

The extended TESLA certificate approach adds additional security to the original TESLA certificate approach, but the performance of the two are equivalent. Compared to public-key authentication, TESLA certificates offer significant savings in power expenditure and the time required to generate authentication parameters for the messages, and to verify the authenticity of messages. As shown in [13], it requires 46 milliseconds to perform a SHA-1 MAC computation on a 4096-bit message using a Pentium-4 2GHz machine, while 2048-bit RSA signing requires 2.26 seconds using the same platform. Therefore the savings is of the order of 49 times when using TESLA certificates as opposed to RSA public-key authentication.

In our algorithm for unicast communication, the source needs to perform one public-key signature verification (in (16)), while the receiver also has to perform one public-key signature verification in (20). Neither the source nor the receiver has to perform public-key signature generation at any time. All the messages from the source to the receiver are authenticated using MACs computed on the messages. Compared to authentication using digital signatures, this represents a substantial savings in computation power and delay. Each message size from the source to the receiver is also smaller in our algorithm compared to using digital signatures. For example, using SHA-1, the authentication MAC for message m is 160 bits, while using 1024-bit RSA key, the signature would be 1024 bits. Therefore, for each message, the authentication field is 6.4 times smaller using TESLA certificates, representing a substantial savings in network bandwidth over a large number of messages.

The savings are similar in the case of multicast communication, where the receivers and the sources have to do public-key signature verification twice on control messages from the CA - one for the CA broadcast in (27), and another time for the initial CA key disclosure broadcast message. The actual data traffic in the group is authenticated entirely using MACs. The use of $GCerts$ helps primarily to reduce the size of messages multicast from the sources to the receivers, since the messages need not contain the source TESLA certificate. Computed over the set of all messages in a group, this represents a significant savings in network bandwidth.

We are currently studying the application of the TESLA certificates to specific network scenarios, and we plan to compute the savings in power, computation and network bandwidth through simulations for the network scenarios we adopt.

B. Difference between TESLA and TESLA Certificate algorithm

One primary difference between TESLA certificates and the original TESLA approach is how bootstrapping is done amongst nodes, such that the anchor element of the source hash chain can be securely distributed to all the receivers. The original TESLA approach requires that all receivers communicate with the source directly for this bootstrapping, and that all receivers either share symmetric keys with the source, or knows the public key of the source. This does not scale well as the network grows, and it is also a problem in networks where the sources and the receivers do not know one another beforehand, or do not have any mechanism to establish the secure channel between them initially. The TESLA certificate approach solves this problem by requiring that all the nodes in the network trust only one network-wide entity, namely, the CA. This greatly simplifies the bootstrapping, since the nodes need to trust only the communication from the CA to verify the anchor element of each source's authentication chain. Also, all the nodes in the network need to synchronize their clocks only with the clock of the CA, which is a simpler approach compared to synchronizing clocks with that of each receiver.

The major drawback with the TESLA certificates is that the CA is a single point of failure. However, many widely-adopted security mechanisms today assume the presence of similar centralized trusted parties, and therefore the CA assumption in TESLA certificates is not an aberration.

C. Advantage of Extended TESLA Certificate Algorithm Over the Original Proposal

The major advantage of extended TESLA certificates over the original proposal of [13], [15] is that it extends the lifetime of each TESLA certificate through the use of source hash chains, the anchor of which is certified by the CA. The frequency of certificate generation by the CA, or the broadcast of key disclosure messages, is reduced as a result. The overhead in the extended approach is that there is an (slight) additional delay between the key disclosure by the CA, and the MAC verification at the receiver - this delay is due to the time it takes for the source to transmit the hash chain element to the receivers.

D. Application of Extended TESLA Certificates

The requirement of an ubiquitous CA restricts the application of the extended TESLA certificate algorithm to networks where there is a trusted party that is reachable by all the nodes in the network. Many network topologies considered at present fall in this category. For example, networks of sensor nodes serviced by a centralized base station would be a good candidate. In this network scenario, the extended TESLA certificate could be applied with minor modifications, with the base station acting as the CA. Since a typical sensor node is considered to be a device of very limited processing, storage and power capabilities, it might be unrealistic to expect a sensor node to perform public-key operations, even if only for the bootstrapping period. In this situation, the functionality of public-key cryptography can be replaced by a shared secret between each sensor node and the base station. Instead of authenticating a source node's certificate using digital signature as in equation (16), the CA would authenticate the certificate by a MAC computed using the secret key it shares with the source node. Similarly, the key disclosure broadcast message from the CA to the network would be authenticated using multiple MACs computed using the individual secret keys shared between the CA and the nodes in the network. This requires pre-shared secrets between the nodes and the base station, which is a common assumption in most sensor networks considered today, where the sensor nodes are deployed in the field with pre-installed keys.

We are currently investigating hybrid network topologies involving terrestrial and space components for a variety of missions, including planetary missions. In our network topologies, there are different types of terrestrial networks consisting of devices of varying capabilities, such as networks of personalized digital assistants, and networks of high-powered sensor nodes. The terrestrial networks are serviced by satellites with a wide broadcast reach that can cover an entire network in a single broadcast. The extended TESLA algorithm is a very good fit for source and message authentication in these network topologies, and we are currently investigating the implementation of TESLA certificates in these network scenarios. We aim to perform simulation studies to see how the extended TESLA certificate algorithm performs in comparison to the original algorithm, and also in comparison to using public-key cryptography.

VIII. Further Research for Extended TESLA Certificate Algorithm

A. Computation of Δ, d

Two parameters that critically affect the security of the extended TESLA certificate algorithm are Δ, the time interval for using a specific element of the hash chain for computing MACs, and d, the key disclosure delay of the CA. Δ is a function of the transmission rate of the source and the reception rate of the receivers, and also the requirements of the security policy that states how frequently the authentication key should be updated.

The value of d is a function of the security policy and depends on the frequency at which the CA broadcasts key disclosure messages. We propose that the CA maintains a consistent d for all the nodes in the network. If d is set at a low value, sources would have a small time window in which to transmit their certificates to the receivers, and get them accepted. Allowing for network delays and other factors, this might lead to situations where legitimate certificates are rejected by the receivers. On the other hand, a large d would be deterimental to the revocation of certificates. Receivers might have to buffer a significant number of packets from potentially revoked sources, before the revocation message can be verified from the next disclosure by the CA.

A detailed analysis of Δ and d, and their inter-dependency, if any, is the subject of future work.

B. Reducing the authentication delay and receiver buffering requirement

The authentication algorithms based on TESLA require that receivers buffer packets till they can be authenticated, due to the delay involved in the source and the CA disclosing their TESLA keys. This buffering requirement leaves the algorithms vulnerable to denial of service attacks, and is a major drawback of the TESLA approach. We are investigating mechanisms that reduce the delay in TESLA key disclosure, and to reduce the buffering requirements at the receivers. Some promising approaches in this regard are the immediate authentication algorithm of [16] and the *staggered TESLA* approach of [13]. We aim to investigate how these mechanisms can be integrated with the extended TESLA algorithm, and whether they can be improved upon in the process.

C. Non-repudiation and bootstrapping issues

As stated in previous sections, non-repudiation is not provided by TESLA certificates. Also, the initial bootstrapping for the nodes require verification of digital signatures. We are investigating additional features to add non-repudiation, and also mechanisms to make the initial bootstrapping process computationally less expensive for the nodes in the network.

IX. Conclusion

In this paper we have re-visited the concept of TESLA certificates, proposed recently in [13], [15]. TESLA certificate is a new type of certificate used for source and message authentication that uses computationally efficient symmetric keys to perform the authentication, instead of public-key cryptography that is used in standard certificates. We have identified several security weaknesses in the original proposal and suggested modifications to strengthen the security properties of the original algorithm. We have proposed a major extension to the TESLA certificate algorithm that is more secure and more efficient than the original proposal. Furthermore, we have introduced the concept of TESLA group certificates, and shown how they can be used to perform efficient source and message authentication in a group setting. As an added advantage, the group certificates can also be used to perform source access control in a multicast group.

However, much work needs to be done to fully validate the algorithms described in this paper. We are currently in the process of implementing the extended TESLA certificate algorithm for a specific network topology consisting of networks of terrestrial sensor nodes serviced by a satellite, with the objective of validating the security of the algorithms and to demonstrate their performance benefits for resource-limited devices. We believe that TESLA certificates hold great promise in allowing efficient authentication for resource-constrained devices where authentication using digital signatures and standard certificates might be too expensive, and this paper will be a useful contribution in this context.

References

[1] H. Krawczyk, M. Bellare, and R. Canetti, *"HMAC: Keyed-Hashing for Message Authentication"*, IETF RFC 2104, February 1997.

[2] N.I.S.T., ""Digital Signature Standard (DSS)"," May 19 1994.

[3] P.R.Zimmermann, *"The official PGP user's guide"*. MIT Press, May 3 1995.

[4] R. Housley, W. Ford, W. Polk, and D. Solo, *"Internet X.509 Public Key Infrastructure Certificate and CRL Profile"*, IETF Network Working Group RFC 2459, http://www.ietf.org/rfc/rfc2459.txt, January 1999.

[5] S. Cheung, ""An Efficient Message Authentication Scheme for Link State Routing"," in *Proc. 13th Annual Computer Security Applications Conference*, San Diego, USA, December 8-12 1997.

[6] R. Gennaro and P. Rohatgi, ""How to Sign Digital Signatures"," in *Advances in Cryptology - CRYPTO '97*. Springer-Verlag Berlin Heidelberg, 1997, pp. 180–197.

[7] R. Anderson, F. Bergadano, B. Crisp, J. Lee, C. Manifavas, and R. Needham, ""A new family of authentication protocols"," *ACM Operating Systems Review*, vol. 32, no. 4, pp. 9–20, 1998.

[8] P. Rohatgi, ""A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication"," in *Proc. Computer and Communications Security Conference (CCS'99)*. Singapore: ACM, 1999.

[9] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, ""Multicast Security: A Taxonomy and Some Efficient Constructions"," *Proceedings of INFOCOMM '99*, March 1999.

[10] A. Perrig, ""The BiBa One-Time Signature and Broadcast Authentication Protocol"," *Proceedings of the 8th ACM Conference on Computer and Communications Security - CCS '01*, November 2001.

[11] A. Weimerskirch and G. Thonet, ""A distributed light-weight authentication model for ad-hoc networks"," in *ICICS 2001*, K. Kim, Ed., vol. LNCS 2288. Springer-Verlag Berlin Heidelberg, 2002, pp. 341–354.

[12] D. Balfanz, D. Smetters, P. Stewart, and H. Wong, ""Talking to strangers: authentication in ad-hoc wireless networks"," in *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)*, 2002.

[13] M. Bohge and W. Trappe, ""TESLA Certificates: An Authentication Tool for Networks of Compute-Constrained Devices"," in *Proc. of 6th international symposium on wirless personal multimedia communications (WPMC '03)*, Yokosuka, Kanagawa, Japan, October 2003.

[14] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, ""The TESLA Broadcast Authentication Protocol"," *RSA Cryptobytes*, Summer 2002.

[15] M. Bohge and W. Trappe, ""An Authentication Framework for Hierarchical Ad Hoc Sensor Networks"," in *Proceedings of the 2003 ACM Workshop on Wireless Security (WiSE'03)*. San Diego, USA: ACM, August 2003, pp. 79–87.

[16] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, ""Efficient and secure source authentication for multicast"," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2001.