Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2011, Article ID 392529, 18 pages
doi:10.1155/2011/392529

Research Article

Energy-Efficient Source Authentication for Secure Group Communication with Low-Powered Smart Devices in Hybrid Wireless/Satellite Networks

Ayan Roy-Chowdhury[1] and John S. Baras[2]

[1] *Institute for Systems Research, University of Maryland, College Park, MD 20742, USA*
[2] *Department of Electrical and Computer Engineering, Institute for Systems Research, University of Maryland, College Park, MD 20742, USA*

Correspondence should be addressed to Ayan Roy-Chowdhury, ayan@umd.edu

Received 1 June 2010; Revised 19 August 2010; Accepted 14 September 2010

Academic Editor: Christos Verikoukis

We describe a new class of lightweight, symmetric-key digital certificates called extended TESLA certificates and a source authentication protocol for wireless group communication that is based on the certificate. The certificate binds the identity of a wireless smart device to the anchor element of its key chain; keys from the chain are used for computing message authentication codes (MACs) on messages sourced by the device. The authentication protocol requires a centralized infrastructure in the network: we describe the protocol in a hybrid wireless network with a satellite overlay interconnecting the wireless devices. The satellite is used as the Certificate Authority (CA) and also acts as the proxy for the senders in disclosing the MAC keys to the receivers. We also design a probabilistic nonrepudiation mechanism that utilizes the satellite's role as the CA and sender proxy. Through analysis, we show that the authentication protocol is secure against malicious adversaries. We also present detailed simulation results that demonstrate that the proposed protocol is much cheaper than traditional public key-based authentication technologies for metrics like processing delay, storage requirements, and energy consumption of the smart devices.

1. Introduction

Large networks of mobile wireless devices have the ability to provide rapid connectivity in disaster areas or military battlefields, or to interconnect users in far-flung geographical locations. However, present limitations on performance, robustness, and security issues have delayed the adoption of such networks. In [1], we have shown that the addition of a satellite overlay to such wireless networks can lead to a great improvement in the network performance. We term this network architecture a *hybrid network*, which has wireless smart devices in terrestrial clusters with dual satellite connectivity providing alternate high-bandwidth and robust forwarding paths through satellite links.

Security is a necessary parameter in hybrid wireless networks if the communication between a pair of smart devices (henceforth also referred to as network *nodes* interchangeably), or a group of devices, is to be protected from unauthorized access. Due to the open nature of the wireless channel, intruders can eavesdrop on the communication between other nodes if the messages are sent in the clear; they can inject fake messages into the network, purporting to come from other nodes, or attempt to modify or delete messages between other nodes. Therefore, strong security mechanisms to prevent such attacks are important, especially for scenarios like military operations where hybrid networks can be of great use.

Many envisioned applications for hybrid networks are collaborative in nature, and it is necessary to ensure that routing control messages and the application data between communicating devices or nodes are properly authenticated to *enable* communication. In this work we therefore focus on source authentication and associated message integrity protocols to facilitate secure communication between groups of wireless smart devices in the field. These security mechanisms are required to prevent attacks against the

network protocols and thereby ensure their correct and robust operation.

Authentication in group communication is traditionally based on asymmetric techniques such as public key-based digital signatures that are appended to the messages [2]. This requires universal access to a trusted Certificate Authority (CA) to generate the certificate binding a node's identity to its public key. Digital signatures also provide *nonrepudiation*—a noncompromised node cannot deny later that it generated a message that has been signed using its private key.

Public key-based authentication is however computationally very expensive (both in CPU cycles and energy expenditure) to generate digital signatures for messages, and also to verify them [3–6]. The public and private keys are larger in size compared to symmetric keys (e.g., 2048-bit RSA public key against 112-bit equivalent symmetric key [7]); the certificates can also take up significant storage space (e.g., X.509 certificate with 1024-bit RSA key is 1 kilobytes while PGP certificate is 1024 bits). In wireless networks where many devices/nodes might have resource constraints, public key cryptography can become a drawback. For example, handheld wireless devices in hybrid networks can have limited processor power, limited storage capacity, and limited available energy supplied by a battery. Performing digital signature generation and verification frequently will consume significant processor capacity and drain the battery quickly. Therefore in hybrid wireless networks it is preferable to use authentication protocols that are based on symmetric cryptography, which expend less node energy. However, designing authentication protocols for group communication using symmetric cryptography is a significant challenge. The primary difficulty is how to create the asymmetry such that each participant has a unique secret with which to authenticate its messages, while allowing all the receivers the capability for validation.

The problem, therefore, is to design an asymmetric source authentication protocol for group communication with resource-constrained devices, that is efficient in terms of energy needs. As a solution to this problem, we propose a digital certificate construct that uses symmetric cryptographic primitives to achieve asymmetric authentication. The certificate is based on a new class of certificates called *TESLA Certificate*. The TESLA certificate concept was originally proposed in [8]. We modify and enhance the original TESLA certificate design and apply the new certificate to hybrid wireless networks to propose an energy-efficient source authentication protocol for nodes in group communication that takes advantage of the centralized infrastructure present in the network, which is the satellite overlay in this particular example of the hybrid network. In the proposed protocol, source authentication using TESLA certificate is based on MAC computation using keyed hash functions, with delayed disclosure of the key by the Certificate Authority (CA), to achieve the asymmetry required for authentication. Due to the use of MACs to generate and verify certificates, the scheme is fast, has low processing overhead, and consumes much less energy than digital signature algorithms. It also avoids the assumption that the user nodes have some sort of security association established a priori, as many other protocols assume. Using the centralized satellite infrastructure, we also design a probabilistic nonrepudiation algorithm for the source authentication protocol.

We refer to our proposed modifications to the TESLA certificate, as the *extended TESLA certificate*. The extended TESLA certificate and the source authentication protocol that is based on it, have been described briefly in [9], while the proposed nonrepudiation algorithm has appeared in [10]. In this paper, we provide an integrated and more extensively detailed description of the proposed algorithms, and add the previously unreported algorithm for key disclosure delay. We also provide more in-depth evaluation of the proposed design, including a detailed security analysis of the proposed protocol and extensive simulation results with detailed analyses. The simulation results demonstrate how much energy efficient the protocol is, in comparison to public key based technologies.

The rest of this paper is organized as follows. We review related work in source authentication algorithms for group communication in Section 2. In Section 3, we describe the TESLA broadcast authentication protocol on which the TESLA certificate is based. The original TESLA certificate algorithm is reviewed in Section 4. We describe our modifications to the original TESLA certificate, and the source authentication protocol based on the extended certificate, in Section 5. The associated probabilistic nonrepudiation protocol is described in Section 6. Security analysis of the proposed protocol is in Section 7. Performance analysis of the authentication protocol is given in Section 8, along with detailed simulation results. We conclude with a brief discussion in Section 9.

2. Related Work

There has been significant research on efficient multicast source authentication algorithms based on symmetric cryptography that attempt to minimize the computation expense of the devices. In the following paragraphs we highlight some of the better known proposals.

Canetti et al. [11] proposed one of the early solutions to use symmetric MACs for multicast source authentication. In their scheme, the source has l keys and computes l MACs on each packet. Each recipient holds a subset of the l keys, and verifies the MAC according to the keys it holds. The authentication protocol has probabilistic security—the choice of key subsets held by each recipient is critical in insuring that with high probability no coalition of up to w colluding members know all the keys held by a good member (where w is the security parameter), and thus maintaining the security of the scheme. The scheme also requires the multicast group members to store a large number of keys.

Gennaro and Rohatgi [12] have proposed a method known as stream signing, where one regular digital signature is transmitted at the beginning of a stream, and each packet either contains a cryptographic hash of the next packet, or a one-time public key using which the one-time signature on the next packet can be verified. However, this approach requires reliable packet transmission, since the loss

of even one packet means that the information required to authenticate future packets will be lost. For most multicast protocols, such reliability cannot be guaranteed, since the transmission protocol is UDP, which is best effort.

Wong and Lam [13] have proposed an approach where the source is allowed to delay and group together several consecutive packets. The source collects the packets in a time interval into an authentication tree and signs the root of the tree. The root signature and hash information on the nodes of the tree are included in each transmitted packet. The signing and verification operations are thus amortized over many packets, and the protocol operations are one to two orders of magnitude faster compared to individual packet signatures.

Rohatgi has proposed a hybrid scheme [14] using offline/online signature generation scheme for creating k-time public/private key pairs so that the cost of signature generation can be amortized over k signatures. The size of the keys is reduced by using hash functions with target collision resistance. The size overhead of the proposed scheme is however still considerable on a per-packet basis (of the order of 300 bytes per packet).

Anderson et al. have proposed the Guy Fawkes protocol in [15], which achieves source authentication using a small number of hash computations. In this protocol, the source selects a series of one-time secrets $X_0, X_1, X_2, \ldots$; the source commits to X_i in message M_{i-1} and reveals it in message M_{i+1}. The commitment for X_i has the form $h(M_{i+1}, h(X_{i+1}), X_i)$, while the first secret X_0 is committed by some external mechanism such as a conventional digital signature. In the Guy Fawkes protocol, the secrets are not related to one another, and the authentication mechanism cannot tolerate packet losses—if a commitment is lost, the corresponding secret cannot be authenticated.

Perrig has proposed a broadcast authentication scheme named BiBa [16] which exploits the birthday paradox in trying to find two or more colliding hash computations on a given message, where the hash values are computed using a set of self-authenticating values (SEALs) $s_1, \ldots, s_t$. The two or more SEALs for which the hash on the message collide form the signature. The scheme exploits the asymmetric property that the source has more SEALs than the adversary, and hence it can easily generate the BiBa signature with high probability. However, the adversary only knows the few SEALs disclosed by the source, and hence has a low probability of forging a valid BiBa signature. The algorithm is probabilistic in nature in the signature generation, and has a significant computation overhead at the source to find a valid signature. Also, the probability of an adversary to forge a signature increases with time as more and more SEALs are disclosed by the source.

As the description in the following sections make clear, our proposed authentication algorithm differs significantly from the ones highlighted in this section. We introduce a new type of certificate, which none of the above attempt to do. The most closely related existing work in this regard is the initial work on TESLA certificates by Bohge and Trappe [8], which we describe separately in Section 4, due to its relation to our design. Also unlike the aforementioned algorithms, our design uses time to provide the asymmetry which makes the individual source authentication in groups possible. The use of time is due to the underlying TESLA broadcast authentication protocol [17, 18], which is reviewed in Section 3.

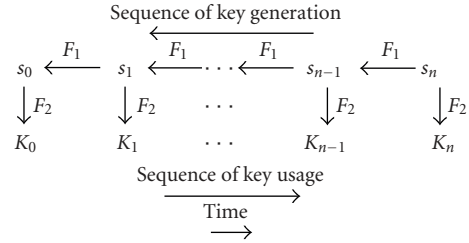


Figure 1: TESLA key generation.

Some of the related work described in this section uses public key-based digital signatures for most authentication, with the cost of signature processing amortized over several packets. In contrast, our protocol does not need public key-based digital signatures, except for the initial bootstrapping phase. In addition, our design is robust to packet losses during transmission such that the authentication of individual packets are not dependent on reliable reception of previous or future packets in the transmission. Moreover, the receivers have to process only one MAC per packet for authentication, where the MAC is computed only on the contents of the associated packet. Either of the former two cannot be claimed of some of the protocols described earlier.

3. Review of TESLA Authentication Protocol

The TESLA broadcast authentication protocol [17, 18] 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.

The TESLA algorithm is illustrated in Figure 1. 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 public one-way function F_1 to s_n. The number of elements of the hash chain correspond to the number of intervals in which the source transmits. The source computes the key used for generating the MAC in each time interval by applying a second public one-way function F_2 to each element of the hash chain.

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.

For each packet generated in time slot i, the source uses the authentication key K_i to compute a MAC on the packet. 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. 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. 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 above is a basic description of TESLA. The algorithm has several enhancements to mitigate various drawbacks; they are described in [18].

4. Review of the TESLA Certificate Algorithm

The idea of certificates based on TESLA was proposed in [17]. The idea has been formalized to form a TESLA-based public key infrastructure (PKI) in [8].

In the algorithm described in [8], there is a certificate authority CA who creates certificates for an entity 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 $\text{Cert}_{\text{CA}_n}(B)$ 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 certificate:

$$\text{Cert}_{\text{CA}_n}(B) = \left(\text{ID}_B, \{aK_{B_n}\}_{tK_{\text{CA}_n}}, n+d, \text{MAC}_{tK_{\text{CA}_n}}(\cdots)\right). \tag{1}$$

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 $\text{Cert}_{\text{CA}_n}(B)$ to B along with aK_{B_n}, which is encrypted with key $K_{\text{CA},B}$ that is shared between the CA and B.

In the time interval $\langle n, n+d\rangle$, a low-powered device D sends a request to B for using B's service: $D \rightarrow B$: (request). To authenticate itself to D, B sends an authentication packet containing its certificate and a MAC on the request:

$$B \longrightarrow D : \left(\text{Cert}_{\text{CA}_n}(B), \text{MAC}_{aK_{B_n}}(\text{request})\right). \tag{2}$$

When D receives the authentication message, it checks the timestamp of $\text{Cert}_{\text{CA}_n}(B)$ to make sure it has arrived before time $n + d$. If the certificate is "fresh", D buffers the authentication packet. At time $n + d$, the CA discloses tK_{CA_n}. Upon receiving the key, D verifies $\text{Cert}_{\text{CA}_n}(B)$ by checking the MAC in the certificate using tK_{CA_n}. If the MAC verifies correctly, D obtains aK_{B_n} from the certificate by decrypting with tK_{CA_n}. Subsequently, D checks $\text{MAC}_{aK_{B_n}}(\text{request})$ to verify the authenticity of B. Therefore, D is able to verify the identity of B only if it receives $\text{Cert}_{\text{CA}_n}(B)$ 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 allows a node to add authentication to packets for a *single* period in time. 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 describe an application of TESLA certificates for authentication in hierarchical ad hoc sensor networks in [19]. 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.

5. Extended TESLA Certificate and Source Authentication Protocol

We extend the original TESLA certificate design and propose a new source authentication protocol based on the extended TESLA certificate, by incorporating the following primary modifications:

(i) we extend the lifetime of the TESLA certificate from single use to multiple uses;

(ii) we allow disclosure of source TESLA keys via proxy;

(iii) we add a probabilistic nonrepudiation mechanism to the source authentication protocol.

A detailed description of the extended TESLA certificate and the source authentication protocol are given in the following sections. We start with a statement of the assumptions that we have made for our solution.

5.1. Network Requirements and Security Assumptions. The extended TESLA certificate implementation requires the presence of a Certificate Authority (CA) to generate the certificates. In our source authentication algorithm, the CA broadcasts the TESLA keys of the source nodes to the network at periodic key disclosure intervals. In the exemplary hybrid network topology discussed in this paper, we use the satellite for providing the services of the CA. The reasons for using the satellite as the CA are as follows.

(i) The satellite is a network node that is always available, connected to the entire network, and is physically secure.

(ii) The satellite has higher computing power with onboard processing capability and higher storage compared to terrestrial wireless nodes.

(iii) The energy available to the satellite is renewable via solar power. Therefore the satellite can perform intensive security and network operations without the risk of draining its energy.

Therefore the presence of the satellite allows the implementation of efficient and secure centralized authentication protocols that would have been difficult to implement in terrestrial wireless networks without comparable centralized infrastructure. Hence we consider the satellite as the root CA in our authentication protocol design and assume that it is trusted by all other nodes in the network. In the proposed source authentication protocol, the satellite generates the TESLA certificates for all the terrestrial user nodes, and it acts as the proxy for the terrestrial nodes for disclosing the TESLA MAC keys used by the nodes for authentication and message integrity—instead of the source node, the satellite broadcasts the TESLA keys to the network at regular time intervals. Therefore the TESLA keys reach all the user nodes in one broadcast transmission. This saves the delay in TESLA authentication, and reduces the processing load on the source nodes, and also the network transmission overhead.

In order to describe the operation of the authentication protocol, let us consider, without loss of generality, a group of three wireless nodes A, B, and C, where A sends messages to B and C. Our objective is to design an authentication mechanism that allows B and C to securely authenticate messages from A using a computationally efficient algorithm that expends low node energy. We make the following assumptions about the initial security setup of the network for authentication purposes:

(i) all three nodes have limited energy and processing power, and none has any pre-existing security information about the others;

(ii) the public key $+K_{\mathrm{CA}}$ of the CA is available to all nodes;

(iii) all nodes are time-synchronized with the CA;

(iv) appropriate security policies are in place to allow each node to securely identify itself to the CA during the initial bootstrapping phase, and each node X shares a unique secret key $K_{\mathrm{CA},X}$ with the CA;

(v) one-way functions F_1 and F_2 [20] are publicly available;

(vi) message transmission from A to B and C start at time t_0;

(vii) time is divided into intervals, each of duration Δ.

The three-user network above can be extended to larger networks with multiple sources and receivers; the extended TESLA certificate and the source authentication protocol described in the following sections would be equally applicable to the larger networks.

5.2. Initial Setup: Key Generation by CA and Source Node. At the time of the initial setup, before any messages are transmitted in the network, the CA and all sources generate the keys that each will need for message authentication. The sets of keys are generated using the TESLA algorithm.

The CA uses a TESLA key chain $\{tK_{\mathrm{CA},i}\}$, $i = \{1, \ldots, N\}$ to authenticate the TESLA certificates that it generates for the group sources. The CA generates a random seed $s_{\mathrm{CA},N}$ and applies one-way function F_1 to $s_{\mathrm{CA},N}$ to form a *hash chain* (3):

$$s_{\mathrm{CA},0} \overset{F_1}{\leftarrow} s_{\mathrm{CA},1} \overset{F_1}{\leftarrow} \cdots \overset{F_1}{\leftarrow} s_{\mathrm{CA},N-1} \overset{F_1}{\leftarrow} s_{\mathrm{CA},N}, \tag{3}$$

where $N > 0$ is equal to the number of unique MAC keys that the CA expects to use for authenticating the certificates and messages it generates during its operational lifetime. The value N depends on the length of each time interval and the total duration that the CA node will perform the function of the CA. We assume that in each time interval, the CA uses only one key for computing the MACs on all the messages it generates in that time interval. Therefore, if the CA's operational lifetime is T and the interval for key disclosure is d, we have $N = T/d$.

It is not necessary that T (and hence, N) are associated with the *entire* lifetime of the CA. T could be a duration that is less than the CA operational life. When time T has elapsed, and the CA has disclosed $s_{\mathrm{CA},N}$, the CA can start with a new hash chain with starting element $N' = T'/d'$ corresponding to a new duration T' and a new disclosure interval d' (which can be the same as d). The CA will have to broadcast the anchor element of the new chain to the network in the manner described in (5), or the new chain can be generated such that one can securely derive $s_{\mathrm{CA},N}$ from the new one. We describe the rest of the algorithms assuming that T is equal to the CA lifetime, since this is not central to the algorithm development.

Once the hash chain has been generated as described in (3), the CA applies function F_2 to each element of the chain to obtain the certificate keys $tK_{\mathrm{CA},i}$, which it uses in the certificates (4):

$$\begin{array}{ccccccccc} s_{\mathrm{CA},0} & \overset{F_1}{\longleftarrow} & s_{\mathrm{CA},1} & \overset{F_1}{\longleftarrow} & \cdots & \overset{F_1}{\longleftarrow} & s_{\mathrm{CA},N-1} & \overset{F_1}{\longleftarrow} & s_{\mathrm{CA},N} \\ & & \downarrow F_2 & & \cdots & & \downarrow F_2 & & \downarrow F_2 \\ & & tK_{\mathrm{CA},1} & & \cdots & & tK_{\mathrm{CA},N-1} & & tK_{\mathrm{CA},N} \end{array} \tag{4}$$

$s_{\mathrm{CA},0}$ is the *anchor element* of the CA's authentication key chain. All TESLA certificates and signed messages from the CA are authenticated using the anchor element during the protocol run. $s_{\mathrm{CA},0}$ is broadcast to the network at time $t < t_0$ (5):

$$\mathrm{CA} \longrightarrow \mathrm{network} : (s_{\mathrm{CA},0}, \mathrm{SIGN}_{-K_{\mathrm{CA}}}(\cdots)). \tag{5}$$

The anchor element itself is authenticated using traditional public key cryptography: the CA generates a signature on the message containing the anchor element and broadcast the message with the signature. All network nodes receiving the broadcasts message verify the signature on the message using the public key $+K_{\mathrm{CA}}$ of the CA. If the signature is verified, the nodes store in local memory the key $s_{\mathrm{CA},0}$ along with the broadcast message.

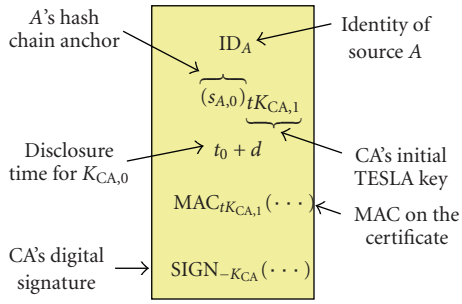


Figure 2: TESLA certificate for node A.

In a manner similar to the above, each 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*, before any messages are sent. A subsequently applies F_2 to each key $s_{A,i}$ generated above and obtains the output $s'_{A,i}$ (3):

$$\begin{array}{ccccccccc} s_{A,0} & \xleftarrow{F_1} & s_{A,1} & \xleftarrow{F_1} & \cdots & \xleftarrow{F_1} & s_{A,n-1} & \xleftarrow{F_1} & s_{A,n} \\ \downarrow F_2 & & \downarrow F_2 & & \cdots & & \downarrow F_2 & & \downarrow F_2 \\ s'_{A,0} & & s'_{A,1} & & \cdots & & s'_{A,n-1} & & s'_{A,n} \end{array} \quad (6)$$

Here $n > 0$ is equal to the number of unique MAC keys that A expects to use for authenticating its messages. The value n depends on the length of each time interval and the total duration of A's transmission. We assume that in each time interval Δ, a source uses only one key for computing the MACs on all the messages it generates in that time interval Therefore, if the total time of A's transmission is T, we have $n = T/\Delta$.

At time $t < t_0$, A sends $s_{A,n}$, n to the CA, along with details on A's key disclosure interval. The message from A to the CA is secured using the shared secret $K_{\mathrm{CA},A}$ between A and the CA. The CA can obtain all the elements of A's TESLA key chain from $s_{A,n}$ and n, as in (3).

5.3. Extended TESLA Certificate. On successful verification of node A's identity subsequent to the steps described in Section 5.2, the CA generates the TESLA certificate for A. A schematic of the TESLA certificate for A is given in Figure 2. The certificate maps the identity of the source node A to the key $s_{A,0}$ which is the anchor element of A's key chain. $s_{A,0}$ is encrypted using key $tK_{\mathrm{CA},1}$ from the CA's key chain. $tK_{\mathrm{CA},1}$ is the CA MAC key for the time period $\langle t_0, t_0 + d\rangle$. The certificate mentions the time $t_0 + d$ up to which the certificate is valid, that is, after time $t_0 + d$, key $s_{A,0}$ is made public to the group and it can no longer be used for new messages. The certificate contains a MAC for authentication, computed on the previous elements using $tK_{\mathrm{CA},1}$.

For added security, the certificate might also contain CA's public key signature on all the previous elements as shown in (7). However, this is not necessary if the CA has previously successfully broadcast the anchor element of its own key chain, following (5):

$$\begin{aligned} &\mathrm{Cert}_{\mathrm{CA}}(A) \\ &= \left(\mathrm{ID}_A, \{s_{A,0}\}_{tK_{\mathrm{CA},1}}, t_0 + d, \mathrm{MAC}_{tK_{\mathrm{CA},1}}(\cdots), \mathrm{SIGN}_{-K_{\mathrm{CA}}}(\cdots)\right), \end{aligned} \quad (7)$$

$$\mathrm{CA} \longrightarrow A : \mathrm{Cert}_{\mathrm{CA}}(A). \quad (8)$$

Here $d \geq \Delta$ is the key disclosure delay for the CA TESLA signature key.

5.4. Message Transmission from Source to Receiver. A sends messages to B and C 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 $\mathrm{Cert}_{\mathrm{CA}}(A)$ with the message:

$$A \longrightarrow \{B, C\} : \left\{M_0 \mid M_0 : \left(m_0, \mathrm{MAC}_{s'_{A,0}}(m_0), \mathrm{Cert}_{\mathrm{CA}}(A)\right)\right\}. \quad (9)$$

Each of B and C checks the *freshness* of the certificate by checking the timestamp of $\mathrm{Cert}_{\mathrm{CA}}(A)$ to make sure it has arrived within the period $\langle t, t_0 + d\rangle$. The receivers also check that $s'_{A,0}$ is not publicly known, that is, $\mathrm{MAC}_{s'_{A,0}}(m_0)$ cannot yet be computed by them. If all the checks pass, B and C store M_0 in their respective buffers, else they discard the message.

Checking the timestamp on $\mathrm{Cert}_{\mathrm{CA}}(A)$ is critical for the security of the algorithm. Once the CA discloses $s_{\mathrm{CA},1}$ at time $t_1 \gtrsim t_0 + d$, any node in the network can create a fake certificate with timestamp $t_0 + d$, allegedly generated by the CA, similar to (7). Therefore receivers will only accept certificates for which the CA TESLA key has not been disclosed at the time of receiving the certificate.

5.5. Message Authentication at Receiver. At time $t_1 = t_0 + d$, the CA broadcasts the key $s_{\mathrm{CA},1}$ to the network:

$$\mathrm{CA} \longrightarrow \mathrm{network} : (\langle t_0, t_0 + d\rangle, s_{\mathrm{CA},1}, \mathrm{SIGN}_{-K_{\mathrm{CA}}}(\cdots)). \quad (10)$$

If receiver B or C has received the anchor element $s_{\mathrm{CA},0}$ (5), they can check the authenticity of $s_{\mathrm{CA},1}$ by verifying $s_{\mathrm{CA},1}$ against $s_{\mathrm{CA},0}$:

$$s_{\mathrm{CA},1} \xrightarrow{F_1} s_{\mathrm{CA},0}. \quad (11)$$

Otherwise, B or C can verify $s_{\mathrm{CA},1}$ from the signature using $+K_{\mathrm{CA}}$. If verification is successful, each receiver derives $tK_{\mathrm{CA},1}$ from $s_{\mathrm{CA},1}$ (4) and uses $tK_{\mathrm{CA},1}$ to verify the MAC on $\mathrm{Cert}_{\mathrm{CA}}(A)$. If the MAC is correct, receiver B obtains $s_{A,0}$ from $\mathrm{Cert}_{\mathrm{CA}}(A)$ by decrypting with $tK_{\mathrm{CA},1}$. B obtains $s'_{A,0}$ from $s_{A,0}$ (6). Then B checks $\mathrm{MAC}_{s'_{A,0}}(m_0)$ using $s'_{A,0}$ and accepts m_0 if the MAC verifies correctly. B saves $\mathrm{Cert}_{\mathrm{CA}}(A)$ and the anchor element $s_{A,0}$ of A's key chain in long-term memory—they are used for authenticating future keys and messages from A.

Messages from A to B in subsequent time intervals use the corresponding key of A's key chain to compute the MAC.

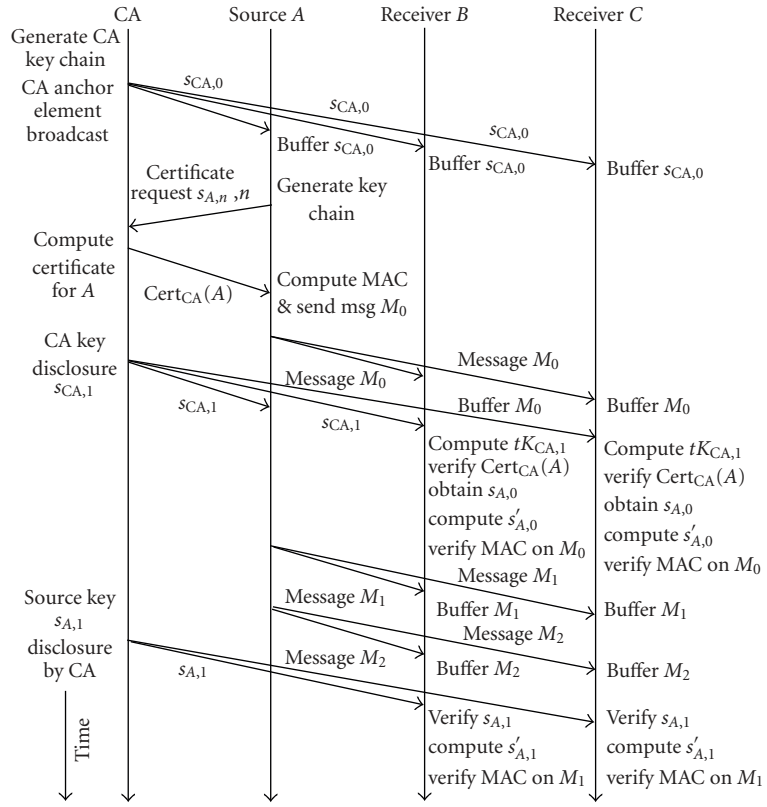


Figure 3: Time diagram for source authentication using the extended TESLA certificate.

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

$$A \longrightarrow B : \left\{M_i \mid M_i : \left(m_i, \text{MAC}_{s'_{A,i}}(m_i)\right)\right\}. \tag{12}$$

At time $t_i + d$, *the* CA *broadcasts* $s_{A,i}$ *to the network*. Since $d > \Delta$, when $s_{A,i}$ is disclosed, A is no longer using $s'_{A,i}$ for computing the MACs on its messages. Any receiver B that receives the CA broadcast, verifies that $s_{A,i}$ indeed belongs to A's MAC key chain as

$$s_{A,i} \xrightarrow{F_1} s_{A,i-1} \xrightarrow{F_1} \cdots \xrightarrow{F_1} s_{A,0}. \tag{13}$$

The above verification is correct since F_1 is a secure one-way function and $s_{A,0}$ has already been verified from $\text{Cert}_{CA}(A)$. However, if B wants to be additionally careful, it can verify $s_{A,i}$ going through the additional steps described above, using the CA key broadcast message and $\text{Cert}_{CA}(A)$. Figure 3 gives a timing diagram representation of the protocol.

After the initial anchor element broadcast 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 $s_{CA,i}$ used in period $\langle t_i, t_i + d\rangle$ at time $t_i + d$. Receiver B can verify that $s_{CA,i}$ belongs to CA's one-way chain:

$$s_{CA,i} \xrightarrow{F_1} s_{CA,i-1} \xrightarrow{F_1} \cdots \xrightarrow{F_1} s_{CA,0}. \tag{14}$$

where $s_{CA,0}$ has been verified before using $+K_{CA}$. B does not need to check CA's signature to verify $s_{CA,i}$.

Thus messages from A to B and C can be authenticated. The source authentication protocol requires that A perform one signature verification to verify the certificate it receives from the CA (7). Each receiver also performs one signature verification on the anchor element broadcast message from the CA (5). Since A is not a receiver, it does not need the verification in (5). All other messages from the CA and the sources can be authenticated using low-computation symmetric MACs. Moreover, sources and receivers do not have to perform clock synchronization directly with one another, synchronizing with the CA is a necessary and sufficient condition for the protocol. This saves additional message rounds and protocol complexity and also breaks the cyclical dependency between authentication and clock synchronization.

5.6. Certificate Revocation. At any time the circumstances warrant that the extended TESLA certificate of a node has to be revoked, the CA would need to broadcast a certificate revocation message to the network. Assume that 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:

$$\text{CA} \longrightarrow \text{network:}\big(\langle t_i, t_i + d\rangle, \text{REVOKE}(\text{Cert}_{\text{CA}}(A)), \text{MAC}_{tK_{\text{CA},i+1}}(\cdots)\big). \tag{15}$$

The receiver buffers the message and waits for the CA to disclose $s_{\text{CA},i+1}$ at time t_i+d. 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. At time $t_{i+1} = t_i + 2d$, the CA broadcasts $s_{\text{CA},i+1}$ to the network. B can verify the authenticity of $s_{\text{CA},i+1}$ from (14) and thus validate the revocation message. 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 looks like

$$\text{CA} \longrightarrow \text{network:}\big(\langle t_i, t_i + d\rangle, \text{REVOKE}(\cdots), s_{\text{CA},i}, \text{MAC}_{tK_{\text{CA},i+1}}(\cdots), \text{SIGN}_{-K_{\text{CA}}}(\cdots)\big), \tag{16}$$

where the REVOKE field will contain the TESLA certificates to be revoked, the MAC is computed on the revoked certificates and the signature verifies $s_{\text{CA},i}$ for nodes that might need the verification (instead of verifying $s_{\text{CA},i}$ using (14)).

Equation (16) implies that the revocation list is sent out at regular intervals with the key disclosure message. The revocation list will contain only those revoked certificates that are otherwise unexpired. Therefore revocation lists in multiple messages might repeat revoked certificates. This helps to make the revocation messages resilient to random channel losses of CA messages.

5.7. Key Disclosure Delay

5.7.1. Time Synchronization. Due to the use of the key disclosure delay to achieve asymmetry in authentication, time synchronization between the nodes taking part in the group communication is important for the correct operation of the source authentication protocol. We use the satellite as the time reference due to its centralized location and global reach. Each terrestrial node synchronizes its local clock with the satellite time. Figure 4 illustrates the time synchronization of the terrestrial nodes with the satellite/CA. The time synchronization algorithm works as follows.

(1) At periodic intervals, the CA broadcasts its local time t_{CA} to the network, authenticated with a digital signature. As shown in Figure 4, let the local times at that instant at a terrestrial source node S and receiver node R be t_1 and t_2, respectively.

(2) Sender S receives the CA time broadcast at its local time t_S. The sender computes the maximum difference in time from the CA as

$$t_S - t_{\text{CA}} = t_S - t_1 + t_1 - t_{\text{CA}} = \delta_S + \epsilon_S, \tag{17}$$

where ϵ_S is the synchronization error of the source clock with the time reference. Therefore the upper bound on the CA's local time, with reference to S, is

$$t \le t_S + \delta_S + \epsilon_S, \tag{18}$$

(3) Receiver R receives the CA time broadcast at its local time t_R. The receiver computes the maximum difference in time from the CA as

$$t_R - t_{\text{CA}} = t_R - t_2 + t_2 - t_{\text{CA}} = \delta_R + \epsilon_R. \tag{19}$$

where ϵ_R is the synchronization error of the source clock with the time reference. Therefore the upper bound on the CA's local time, with reference to R, is

$$t \le t_R + \delta_R + \epsilon_R. \tag{20}$$

The above method of time synchronization with the CA also indirectly synchronizes the time between the terrestrial nodes. After synchronization, the difference in time between the nodes S and R is

$$\delta_S - \delta_R + \epsilon_S - \epsilon_R. \tag{21}$$

It can be deduced from Figure 4 that δ_S and δ_R are the network propagation times from the satellite to the terrestrial nodes, that is, $\delta_S \approx \delta_R$ and hence the difference in local time is only

$$\epsilon_S - \epsilon_R = \epsilon_{SR} \tag{22}$$

If S and R are well synchronized with the CA: $\epsilon_{SR} \approx 0$.

5.7.2. Computation of the Key Disclosure Delay. The key disclosure delay d is a critical parameter affecting both the security and the performance of the proposed protocol. It depends on the duration of each time interval Δ and the network propagation delay from the sources to the receivers. As is shown below, if the message transmission from the sources to the receivers happens exclusively over the satellite links, then the key disclosure delay depends only on Δ and the satellite link propagation delay, which is known and fixed.

We follow the method outlined in [18] in computing the key disclosure delay for our proposed protocol. Let us consider a packet p_j being sent by source S and received by node R in time interval I_j. Let the local times at S be t_j^S when the packet is sent, while the local time at R is t_j^R, when the packet is received. The requirement to satisfy the security condition is

$$\left\lfloor \frac{t_j^R + \delta_R + \epsilon_R - t_0}{\Delta} \right\rfloor - I_j < d, \tag{23}$$

where δ_R and ϵ_R are the propagation delay from the CA to the receiver and the time synchronization error at R, respectively, as explained in Section 5.7 and shown in Figure 4

Also, we must have

$$t_j^S < t_0 + I_j * \Delta + \Delta. \tag{24}$$

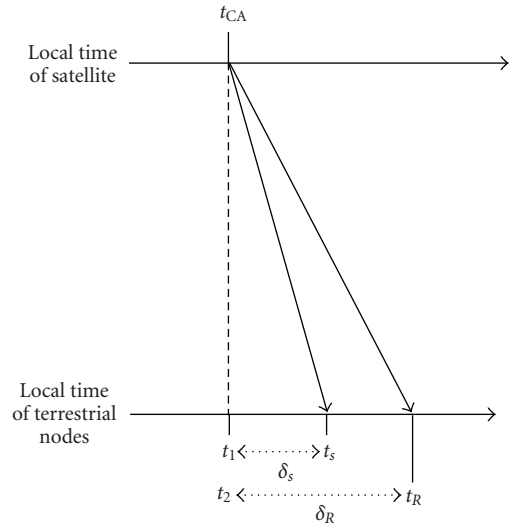


Figure 4: Time synchronization between the protocol participants in the extended TESLA certificate protocol.

Equation (24) merely states that the local time at the source must correspond to the time in interval I_j. If D_{SR} is the network propagation delay from S to R, then

$$D_{SR} = t_j^R + \epsilon_R - t_j^S - \epsilon_S, \tag{25}$$

where ϵ_S is the time synchronization error for S.

From (23), (24), (25), we get:

$$\left\lfloor \frac{D_{SR} - \epsilon_R + t_j^S + \epsilon_S + \delta_R + \epsilon_R - t_0}{\Delta} \right\rfloor - I_j < d,$$

$$\text{that is,} \quad \left\lfloor \frac{D_{SR} + t_0 + I_j * \Delta + \Delta + \epsilon_S + \delta_R - t_0}{\Delta} \right\rfloor - I_j < d,$$

$$\text{that is,} \quad \left\lceil \frac{D_{SR} + \epsilon_S + \delta_R}{\Delta} \right\rceil + 1 < d. \tag{26}$$

If we assume that the packet transmission from S to R is over the satellite links, then D_{SR} is one-way propagation delay over the satellite from S to R, while δ_R is the direct propagation delay from the satellite to any terrestrial node. We thus have $D_{SR} = 2 * \delta_R$ and hence from (26), we get

$$\left\lceil \frac{3 * \delta_R + \epsilon_S}{\Delta} \right\rceil + 1 < d. \tag{27}$$

For a given satellite configuration, δ_R is known and fixed. For example, for a geostationary satellite, δ_R is of the order of 0.12 seconds. Moreover, for fine synchronization of the terrestrial nodes with the time reference, ϵ_S is negligible compared to the satellite propagation delay. Hence, (27) gives

$$\left\lceil \frac{0.36}{\Delta} \right\rceil + 1 < d. \tag{28}$$

Figure 5 shows the variation in the key disclosure delay d as a function of the key use interval Δ. As the figure demonstrates, the product of the disclosure delay and the key use interval is bounded by the network propagation delay, which is a constant. Hence as the key use interval increases from 1 ms to up to 50 ms, the disclosure delay varies between 361 ms and 410 ms.

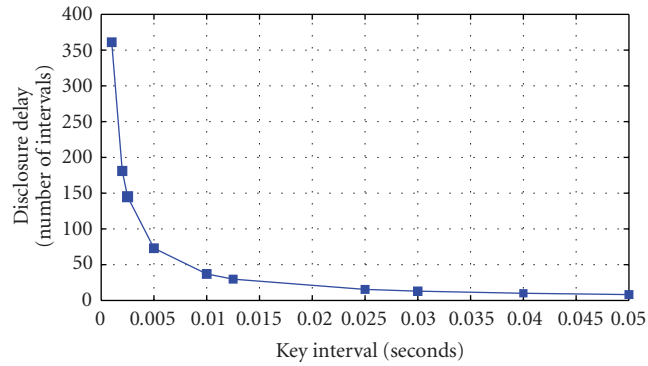


Figure 5: Key disclosure delay d as a function of key use interval Δ.

6. Nonrepudiation of the Source Authentication Protocol

Nonrepudiation is not provided by the TESLA authentication algorithm [17] or the original TESLA certificate proposal [19]. The symmetric nature of the basic cryptographic primitive used here—MACs—does not allow for nonrepudiation. Once the hash key for a particular MAC is disclosed, any group member would be able to generate the MAC for the given message. Therefore, at a later instant in time, it is impossible to prove that the message was generated by a particular source. The lack of nonrepudiation is a major drawback of the previous TESLA-based source authentication algorithms, compared to public key-based authentication.

In our extended TESLA certificate algorithm, we propose to add nonrepudiation by taking advantage of the satellite infrastructure and the proposed mechanism of key disclosure by proxy. This is achieved as follows.

The source authenticates each message by *two or more* MACs, computed using keys from two or more key chains, respectively. The root key of each chain is shared between the source and the CA (the satellite) as described in Section 5.2, and the anchor elements of all the key chains used by the source is included in the extended TESLA certificate for that source. The source includes all the MACs with each message transmission. Each receiver buffers the message along with all the MACs if the basic security check is satisfied, as described in the protocol in Section 5.5. At the time of key disclosure, the CA broadcasts *only one of the MAC keys* out of the set of MAC keys for the given source and message. Each receiver verifies the single MAC associated with the key broadcast by the CA, and accepts the message as correct if the MAC is verified. If any receiver wants to be able to check the message for nonrepudiation at a later time instant, it saves the message along with all its MACs.

The MAC key that is disclosed by the CA is chosen at every disclosure instant, with uniform probability from the set of available keys for that time interval. Therefore, the source cannot know in advance, with a high degree of probability, which key will be used by the receivers for authentication. Hence, if the source would like its messages to be accepted by the receivers, it will have to include all the MACs correctly computed with the corresponding keys.

If at a later instant in time, a receiver would like to prove that a message was indeed generated by the source (i.e., nonrepudiation), the receiver can simply send a nonrepudiation request to the CA. Upon receiving the request, the CA discloses one of the *previously undisclosed* MAC keys for the message in question. The receiver can compute the MAC for the message with the newly disclosed key and compare the MAC with the set of MACs it had saved previously. If the CA and the receiver operates correctly, the newly computed MAC will match one of the saved MACs. Since (i) the undisclosed MAC keys were known only to the source and the CA, and (ii) the CA is universally trusted, therefore the saved MAC must have been computed by the source using its MAC key and hence the message must have been generated by the source. Thus nonrepudiation is achieved.

The security of the above algorithm is proportional to the number of MACs included with each message. For two MACs per message, the probability of a particular key being disclosed by the CA is 0.5. We term this probability the *r-factor*, where r is acronym for repudiation. It is computed as the inverse of the number of MACs included with each message. A nonconforming source which includes only one correctly computed MAC with its message in order to avoid nonrepudiation, can expect the message to be accepted by the receivers only with 50% probability. If four MACs are included with every message, the r-factor is to 0.25, and so on. There is hence a tradeoff between the strength of the nonrepudiation algorithm and the security overhead per message in terms of number of MACs involved. There is also the processing overhead at the source since its node has to compute M, number of MACs per message where $M > 1$.

The number of MACs per message also affects the security of the algorithm in the context of the receivers. If there are two MACs per message, the nonrepudiation mechanism will be successful for the request from one receiver. For any subsequent request from other receivers for that particular message, nonrepudiation will fail since both MAC keys are now known to the receivers. The number of successful nonrepudiation requests for a given message is therefore directly proportional to the number of MACs per message. This drawback can be solved by modifying the protocol steps for nonrepudiation. Instead of sending a request for an undisclosed key, the receiver can send the entire message along with the saved MACs, to the CA. The CA itself will compute the MACs on the message with any one of the undisclosed keys and compare with the saved MACs sent by the receiver. Since the undisclosed keys are known only to the CA and the source, in the event of a match, the CA can confirm to the receiver that the message was indeed generated by the source. The security of this mechanism depends only on the amount of trust placed on the CA and is independent of the number of MACs per message. The tradeoff is the additional load on the CA and the network overhead in transmission of the message with the MACs, to the CA.

7. Security Analysis: Prevention of Authentication Attacks

The source authentication protocol based on the extended TESLA certificate is resistant to active attacks by malicious nodes in the network. In the following sections we discuss the security provided by the protocol against specific active attacks. In our analysis, we assume that the CA is always secure, since compromise of the CA is a single point of failure for security in the network.

7.1. Malicious Node with Connectivity to Source and Receiver. We consider the case where a malicious node X attempts to create fake packets from a source to the receiver(s). Without loss of generality, we consider one source A is sending data to one receiver B, the data being authenticated using the proposed source authentication protocol. We assume that X can hear packet transmissions from A, and can also transmit to B. X can also receive the broadcast messages from the CA. Therefore, shortly after time $t_0 + d$, X has knowledge of $\text{Cert}_{\text{CA}}(A)$, message M_0 from A to B, $s_{\text{CA},0}$ broadcast by the CA, and $s_{A,0}$ from the certificate. X can verify that $s'_{A,0}$ belongs to the authentication hash chain of A by performing the verification procedure. Having obtained a verified element of A's authentication chain, X can attempt to spoof messages as coming from A, starting at time $t_0 + kd$, where $k > 0$. 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 $O(2^K)$, where each element of the hash chain is K bits and K is assumed to be large (for example, K is

160 bits for SHA-1 HMAC [21]). 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 generates an element of A's hash chain without knowledge of any legitimate element of the hash chain. This has the same computational complexity of $O(2^K)$ and is computationally infeasible for any X with finite resources.

A third approach X attempt would be to generate an independent hash chain that produces the hash value $s_{X,0}$ that is computationally indistinguishable from $s_{A,0}$. 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 property of F_1 and F_2.

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 (7), and also generate fake CA key disclosure broadcast message similar to (10). However, unless X knows the CA private key $-K_{\mathrm{CA}}$, it will not be able to correctly sign the fake $\mathrm{Cert}_{\mathrm{CA}}(A)$, 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 $+K_{\mathrm{CA}}$. As per our assumption of the security of the CA, $-K_{\mathrm{CA}}$ 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 (10), but (a) the fake hash element $s_{CA_X,i}$ will not verify successfully to the anchor element $s_{\mathrm{CA},0}$ and (b) this does not allow X to fake elements of A's hash chain.

7.2. Attack on the CA Revocation Messages. A malicious node X in the network can attempt to broadcast fake revocation messages, similar to (15), 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 $s_{\mathrm{CA},i+1}$, with knowledge of at most the key $s_{\mathrm{CA},i}$, where $s_{\mathrm{CA},i+1} = F_1^{-1}\{s_{\mathrm{CA},i}\}$. Using reasoning similar to the previous section, owing to the one-way property of F_1, this has computational complexity $O(2^K)$ 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) $s_{\mathrm{CA},i+1}$, and therefore be discarded.

Therefore, the source authentication protocol based on the extended TESLA certificate approach is secure against message spoofing attacks by malicious nodes in the network.

7.3. Signal Jamming Attacks. Adversarial nodes can attempt to prevent the successful transmission of protocol messages by jamming the channel with unwanted signals. This is a physical layer attack and not an attack against the authentication algorithms per se. Methods to achieve resiliency against such attacks are ideally the domain of the physical and link layer algorithms, for example, spreading algorithms and acknowledgment-retransmission algorithms, amongst other measures.

For the hybrid network architecture described in this work, a large network-wide signal corruption is mostly infeasible, due to the vast footprint of the satellite. Channel losses are localized to small regions at any given time; the satellite physical and link layer protocols incorporate sufficient mechanisms to recover from such losses.

At the same time, the proposed authentication algorithm is robust to random channel losses of protocol messages.

(i) Authentication of any individual packet is independent of authentication of prior packets. If the source key disclosure message containing the keys for a set of buffered packets, is lost due to channel errors, then only that set of buffered packets cannot be authenticated; it does not affect packets whose MAC keys are not in the lost disclosure message.

(ii) If a certificate revocation message (associated with the CA key disclosure message) is lost due to channel errors, the receivers will not be aware of any newly revoked certificate that is contained in the lost message. Consequently the packets from the corresponding source will continue to be treated as valid, till the next revocation message is received, or the certificate expires, whichever is earlier.

8. Performance Evaluation of Extended TESLA Certificate Algorithm

We have run simulations of the proposed source authentication protocol to analyze the demands the protocol can make of node resources, and also to compare it with public key-based authentication protocols. For our performance analysis, we consider that all security protocol and data messages from the source to the receivers are sent over the satellite channel where the satellite is in Ka-band and geostationary orbit. Therefore, the one-way satellite propagation delay is of the order of 130 ms and the terrestrial propagation delays from the group nodes to the local gateways are negligible in comparison. Moreover, the satellite uplink bandwidth is much less in comparison to the satellite downlink bandwidth and usually also lower than the terrestrial wireless bandwidth (assuming the wireless MAC protocol is IEEE 802.11). By our assumption, all data has to traverse the satellite uplink and hence we limit the overall network bandwidth to be equivalent to the satellite uplink bandwidth. In the following analyses, the uplink bandwidth is varied between 64 Kbps and 10 Mbps.

8.1. Size of the Buffer at the Receivers. An important resource consideration is the amount of memory or buffer required in the receivers for temporary storage of the data packets that are pending authentication (that is, the MAC key has not been disclosed by the CA). Figure 6 shows the variation in the size of the buffer required for different Δ and varying network bandwidths.

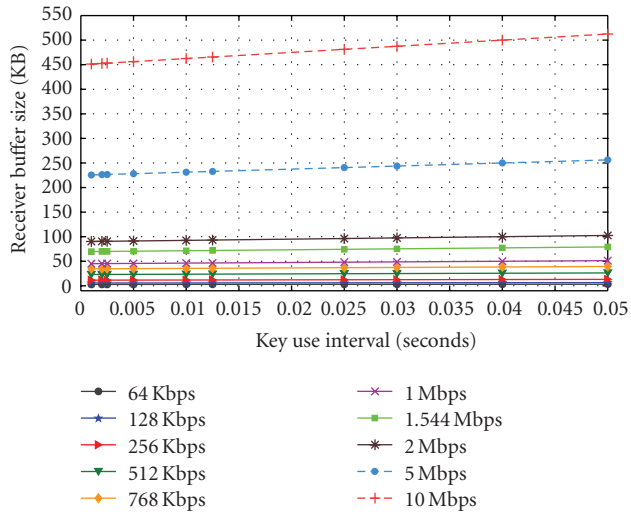


FIGURE 6: Receiver buffer size for varying network bandwidths and key use intervals.

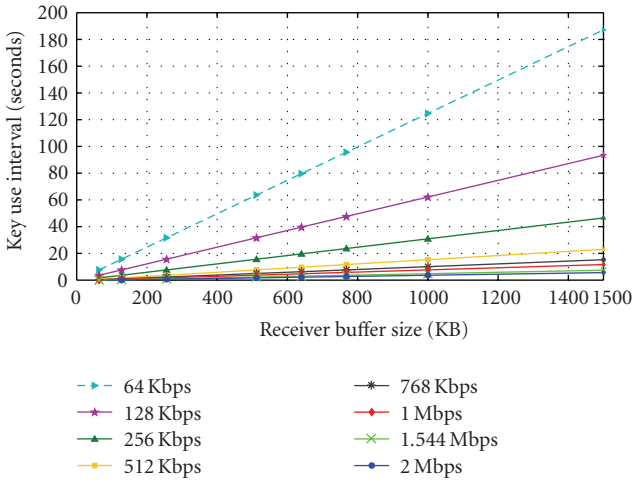


FIGURE 7: Key use interval Δ as a function of receiver buffer size and network bandwidth.

In theory, the receiver buffer size depends on the time interval of key use (since the key disclosure delay is a function of that) and the network bandwidth. As Figure 6 shows, the buffer size varies little with the key disclosure delay since it is limited to a narrow range by the network propagation delay. However, the buffer size is significantly affected by the rate at which data is received—for higher rates, larger buffer is required. Even so, the buffer requirement is only in the order of hundreds of kilobytes, which is a small fraction of the memory present in wireless smart devices today.

If the receiver buffer is fixed due to hardware constraints, the key disclosure delay (and hence the key use interval) has to be dynamically determined based on the smallest buffer size available, and the network bandwidth. Figure 7 shows that the key use interval can be longer for larger buffers, while it is shorter for higher bandwidths.

8.2. Comparison of the Certificate Size. The size of the TESLA certificate and the MACs computed on each message, compare favorably to digital certificates and signatures used in public key-based cryptography. If the MAC algorithm is based on SHA-1 [21], the key used is 160 bits. For a TESLA certificate with fields as shown in 2, the certificate size is computed as follows.

(i) 32 bits for the ID—this will cover 4 billion nodes;

(ii) 160 bits for the encrypted anchor element (128 bits if MD5 is used instead of SHA-1);

(iii) 64 bits for the time field—this gives the current time in UTC since January 1, 1900, with a resolution of 200 picoseconds [22];

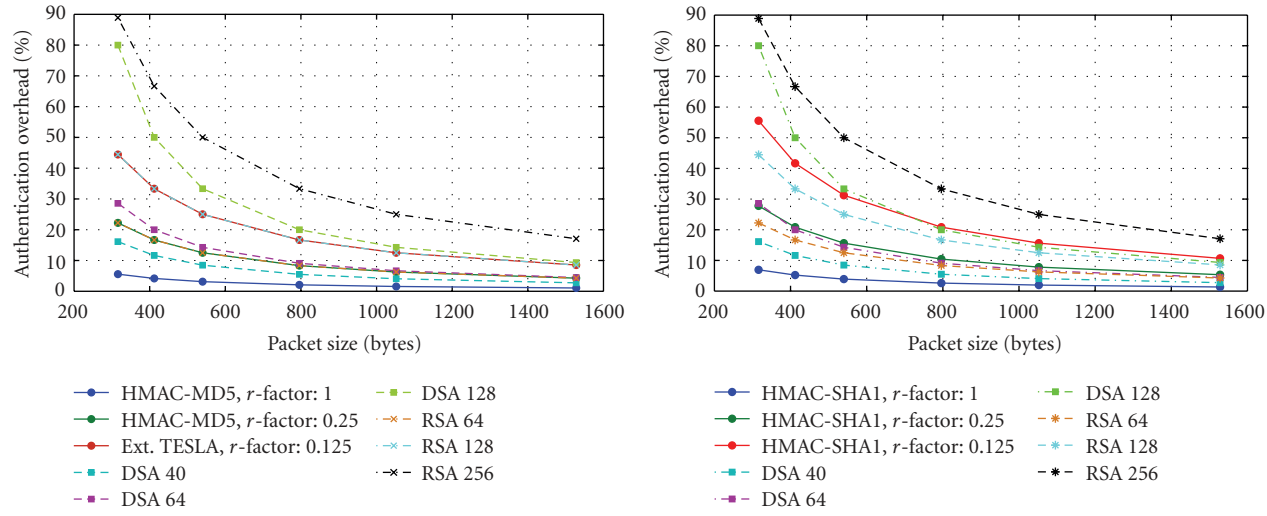
(iv) 160 bits for the MAC;

(v) 1024 bits for the CA digital signature (assuming PKCS no. 1-based [23] digital signature with modulus 1024 bits).

Therefore, the total size of the certificate is 1440 bits or 180 bytes. In contrast, a typical X.509 certificate size is of the order of 1 KB.

8.3. Comparison of Signature Size Overhead. The size of the MAC appended to each message in our protocol is 128 bits for HMAC-MD5 or 160 bits for HMAC-SHA1 (and with r-factor 1). In comparison, if RSA-based digital signature is used, the signature size would be 512 bits, 1024 bits, or 2048 bits for RSA modulus $N = 512, 1024, 2048$, respectively. For DSA or ECDSA-based digital signatures, the signature size is 320 bits for a security level of 80 bits [2].

A comparison of the size overhead incurred for authenticating 500 MB data using our proposed protocol or DSA or RSA, is shown in Figure 8. Figure 8(a) compares extended TESLA with HMAC-MD5 against DSA and RSA, while Figure 8(b) compares extended TESLA with HMAC-SHA1 against DSA and RSA. Since the size of the message is larger than the size of the maximum bytes allowed per transmission, the message is split into smaller chunks for transmission and each individual chunk is authenticated separately. The graphs show how the overhead varies as a percentage of the total bytes transferred (message + MAC/signature) as the size of the IP packet varies between 316 bytes and 1528 bytes. For extended TESLA, we consider three cases based on the degree of nonrepudiation present—for each packet, r-factor is 1 (one MAC), 0.25 (four MACs) or 0.125 (eight MACs). These are compared to DSA with signature sizes 40, 64, and 128 bytes, and RSA with signature sizes 64, 128, and 256 bytes ($N = 512, 1024$, and 2048, resp.). For all the cases, the overhead decreases with increase in the packet size because for the higher packet sizes there are lesser number of chunks and hence lesser number of MACs or signatures. The overhead for our basic protocol is the lowest of all the cases. As we add more MACs for nonrepudiation, the overhead goes up and is a significant percentage for eight MACs (r-factor 0.125). This is the tradeoff in terms of size for our nonrepudiation scheme. However, even then the overhead for our protocol is significantly less than the



(a) HMAC-MD5, DSA, and RSA

(b) HMAC-SHA1, DSA, and RSA

Figure 8: Comparison of percentage byte overhead due to authentication for 500 MB data.

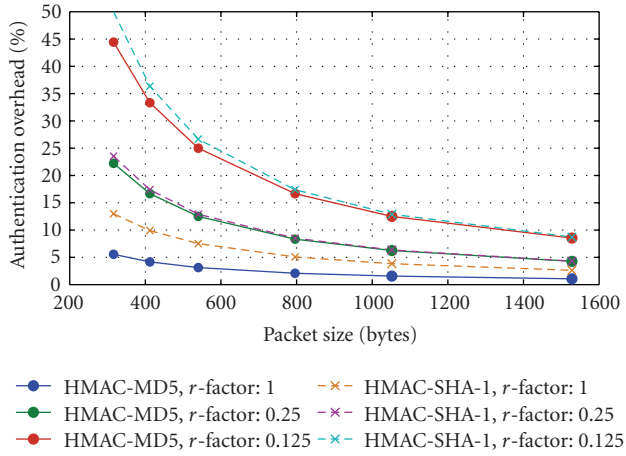


Figure 9: Percentage size overhead comparison between HMAC-MD5 and HMAC-SHA1, 500 MB data.

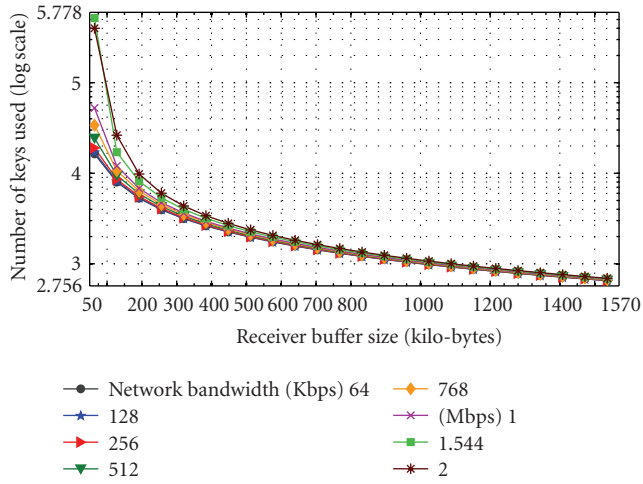
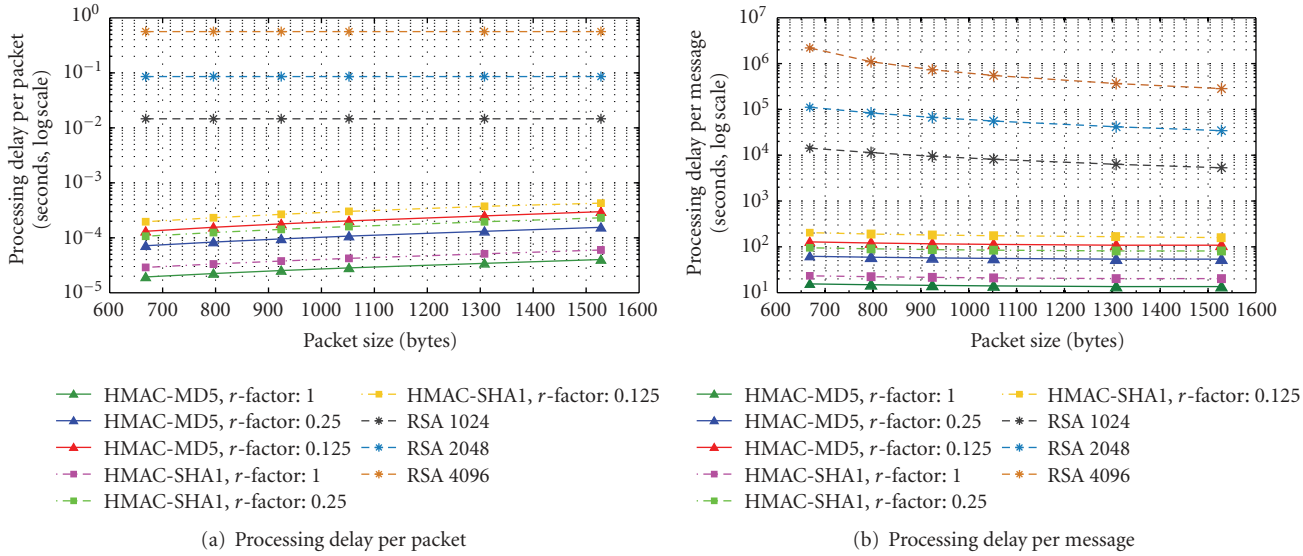


Figure 10: Number of keys required for authenticating 1 GB message.

overhead due to strong RSA (256 byte signature) or DSA (128 byte signature-) based security. The graphs suggest that we should go for the largest packet size allowed by the underlying link layer protocol, since the overhead drops significantly. However, this has to be considered against the energy expense for authenticating larger packet sizes, as shown in some following figures.

Figure 9 shows that the size overhead with HMAC-SHA1 is more than that with HMAC-MD5 for all r-factors, since each individual MAC for the former is 1.25 times in size that of the latter. However, for higher r-factors and larger packet sizes, the overheads tend towards equal values. The graphs again emphasize the need for larger packets, since the overhead drops from 50% for the worst-case scenario (HMAC-SHA1, r-factor 0.125, 316 byte packet), to 8.72% (1528 byte packet).

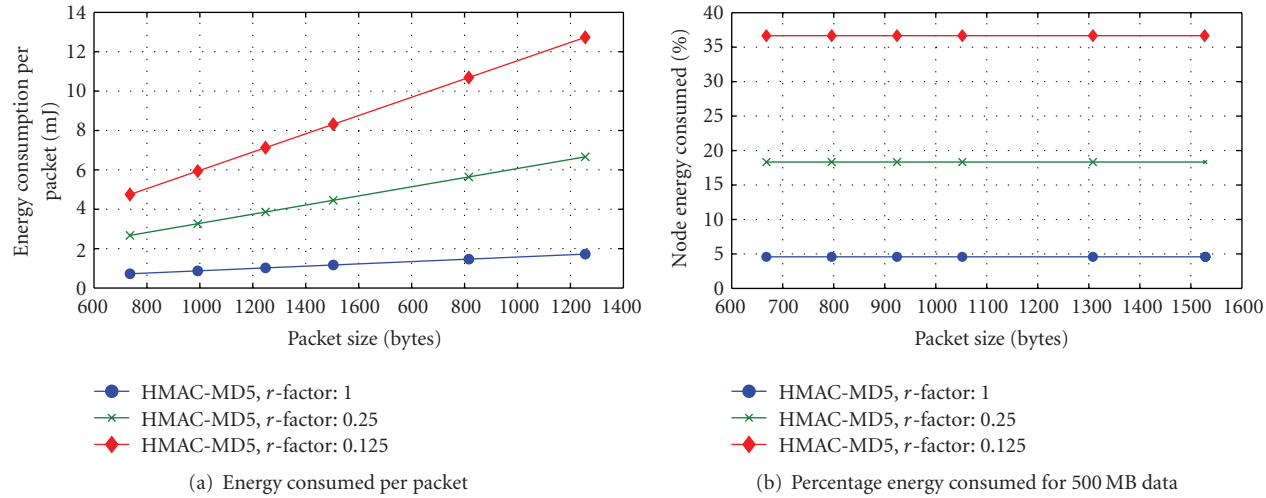
8.4. Number of MAC Keys Required. The number of cryptographic keys required for computing the MAC on each block, for 1 GB data, as a function of the buffer size and the network bandwidth, is shown in Figure 10. Here for each time interval Δ, one key is used. Δ is also lower bounded by 10 ms. The buffer size is varied from 64 KB to 1536 KB in steps of 64 KB. All the graphs exhibit a knee for buffer size 128 KB. The graphs show that for high-transmission rates, the number of keys required is high for small buffers, while it decreases sharply as the buffer size increases. The decrease with increasing buffer sizes is more smooth for bandwidths lower than 1 Mbps. For large buffer sizes, the number of keys required becomes largely independent of the network bandwidth. When the buffer sizes are small, the key use intervals have to be small too, and they are more susceptible to variations in the data rate. For large buffers,



(a) Processing delay per packet

(b) Processing delay per message

Figure 11: Comparison of authentication processing delay for 500 MB data, PIII500 MHz.



(a) Energy consumed per packet

(b) Percentage energy consumed for 500 MB data

Figure 12: Energy consumption for HMAC-MD5 authentication only, iPAQ H3670.

the effect of the bandwidth is less important. Therefore, one should consider the receiver buffer size as a more important factor than the network bandwidth in designing the protocol parameters.

8.5. Comparison of Processing Delay Overhead. An analysis of the processing delay overhead of the extended TESLA protocol, and its comparison to the processing delay for RSA signatures, is given in Figure 11. We simulate the delay due to authentication for 500 MB data on a 500 MHz Pentium III machine. The delay figures for HMAC-MD5 and HMAC-SHA1 are computed based on the approximation that each operation is executed in one processor clock tick for the 500 MHz PIII processor. The delay figures for RSA per packet for the 500 MHz PIII processor are from [24] and are reproduced here in Table 1.

Table 1: RSA signature timings (ms) per packet on 500 MHz Pentium III [24].

Processor	Key length (bits)		
	1024	*2048*	*4096*
PIII-500 MHz	14.6	85.6	562.8

Figure 11(a) validates a key feature of each algorithm compared here. For the HMAC algorithms, the processing delay is proportional to the number of 512-bit blocks being processed. As the packet sizes increase, the number of blocks per packet also increase, and therefore the processing delay for HMAC increases. Also, HMAC-SHA1 performs more

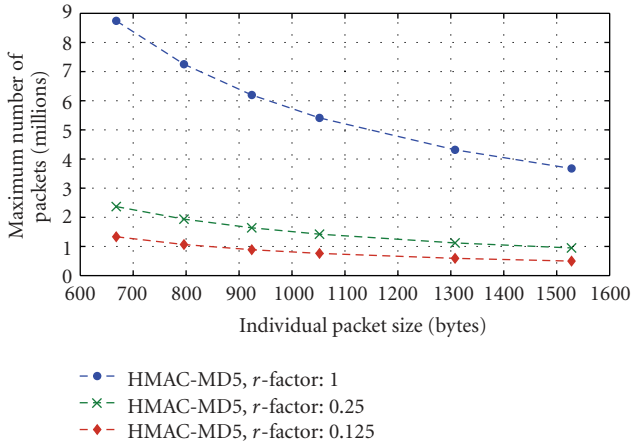


Figure 13: Total number of packets processed by source authentication protocol with HMAC-MD5 on iPAQ H3670.

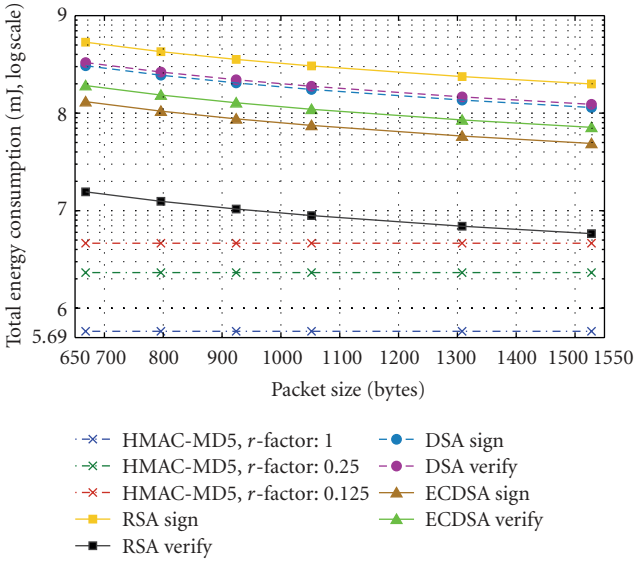


Figure 14: Comparison of total energy required for authenticating 500 MB data on iPAQ H3670.

operations (1110 per 512-bit block) compared to HMAC-MD5 (744 per 512-block) and this fact is reflected in the graphs. For RSA, the processing delay depends only on the size of the modulus N and is independent of the size of the message. The cheapest RSA delay, 14.6 ms per packet (for modulus 1024 bit), is still significantly higher than the most expensive HMAC delay, 0.43 ms (for HMAC-SHA1 with eight MACs). This is a major advantage of our protocol over signature-based schemes.

The total processing delay for authenticating 500 MB data is shown in Figure 11(b). The total number of 512-bit blocks in the overall message is independent of the size of each packet, and hence also the HMAC processing delay for the entire message. However, for RSA the processing delay is directly proportional to the number of message chunks processed. Here also our protocol performs significantly better than using RSA signatures—the worst-case delay is

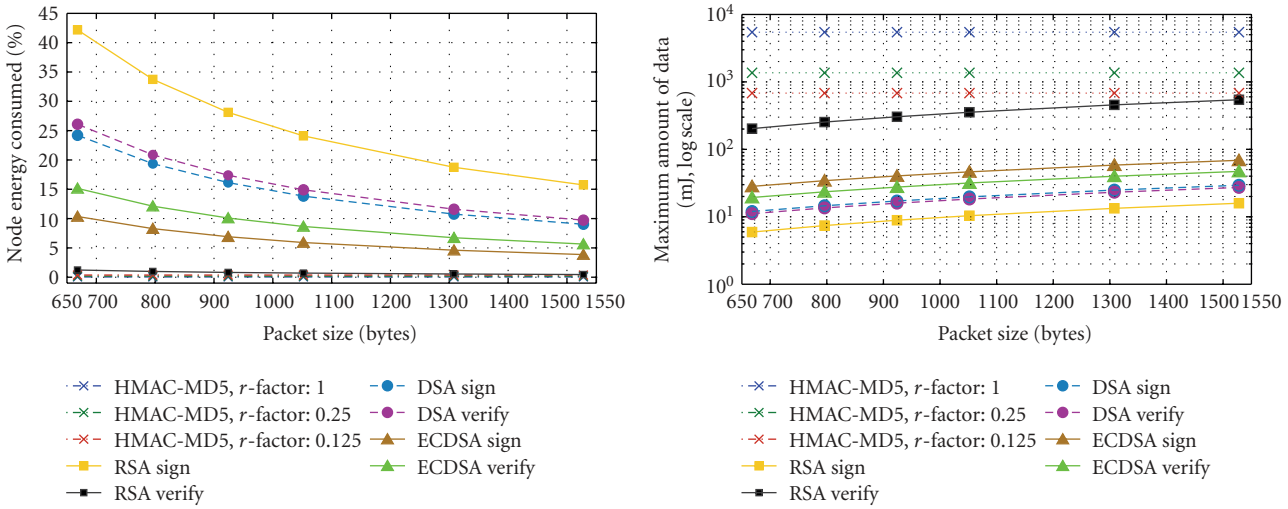
Table 2: Energy cost of digital signature algorithms and HMAC, for Compaq iPAQ H3670 [26].

Algorithm	Key size (bits)	Sign (mJ)	Verify (mJ)
RSA	1024	546.50	15.97
DSA	1024	313.60	338.02
ECDSA	163	134.20	192.23
HMAC		1.16 (μJ/B)	

204 seconds (for HMAC-SHA1, 8 MACs), while the best-case delay for RSA is a prohibitive 5321 seconds (for $N = 1024$, 1528-byte packets).

8.6. Analysis of Energy Consumption. We analyze the energy consumption of our protocol for authenticating 500 MB data in Figure 12. We simulate the energy consumption for a Compaq iPAQ H3670 handheld computer [25], which is fairly representative of the low-power smart devices that we have considered in our protocol design. The handheld contains an Intel SA-1110 StrongARM processor clocked at 206 MHz. It is powered by a Li-polymer battery with capacity 950 mAh at 3.7 V, which gives the total battery capacity to be 12654 Joules. The base figures for energy expenditure of different cryptographic operations of the handheld are obtained from [26] and are reproduced here in Table 2.

Figure 12(a) shows that the energy consumption for authentication each packet ranges between 0.7238 mJ and 12.73 mJ, with larger packets and higher r-factors consuming more energy, as is expected. Figure 12(b) shows the total energy consumed for authenticating 500 MB data, as a percentage of the battery capacity. For higher r-factors, the energy consumption is a significant percentage of the capacity and implies that more than 500 MB data cannot be authenticated without recharging. However, it is to be noted that in most cases, the higher energy expense for the higher r-factors is incurred by the source node only. The receivers can authenticate the messages by computing only one MAC and hence the figures for r-factor 1 is indicative of the energy expense of the receivers. Given the energy constraints, the maximum number of packets that can be authenticated by our protocol, using HMAC-MD5, is given by Figure 13. In this simulation we assume that only 50% of the total energy is used in the protocol operation and the rest for other purposes such as packet transmission. (The 50% upper limit indicates our belief that useful security schemes should not be energy intensive. Most real-life implementations would use far lower energy amounts so that the node can perform meaningful functions other than security operations.) The graphs show that for larger packet sizes, the total number of packets that can be processed are lower, since more energy is spent per packet, due to the higher number of 512-bit blocks. In the worst case, a source with r-factor 0.125 can process between 1.3 million (IP packet size 668 bytes) and 0.49 million (1528 byte per packet) packets. On the other hand, a receiver (with r-factor 1) can process between 8.74 and 3.66 million packets for packet sizes 668 bytes and 1528 bytes, respectively.



(a) Percentage energy consumed for 5 MB data

(b) Maximum amount of data processable

Figure 15: Energy performance comparison of different authentication protocols on iPAQ H3670.

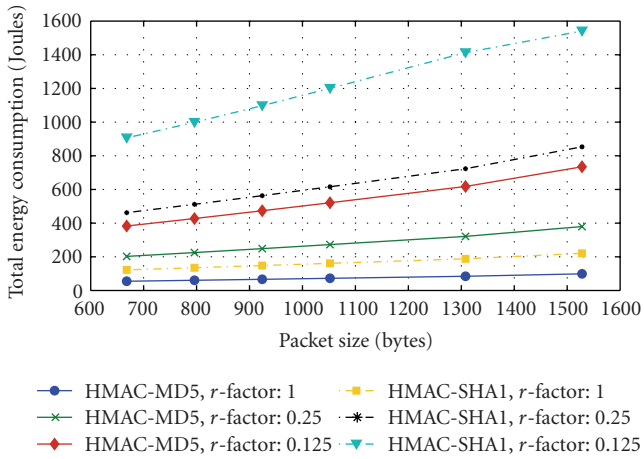
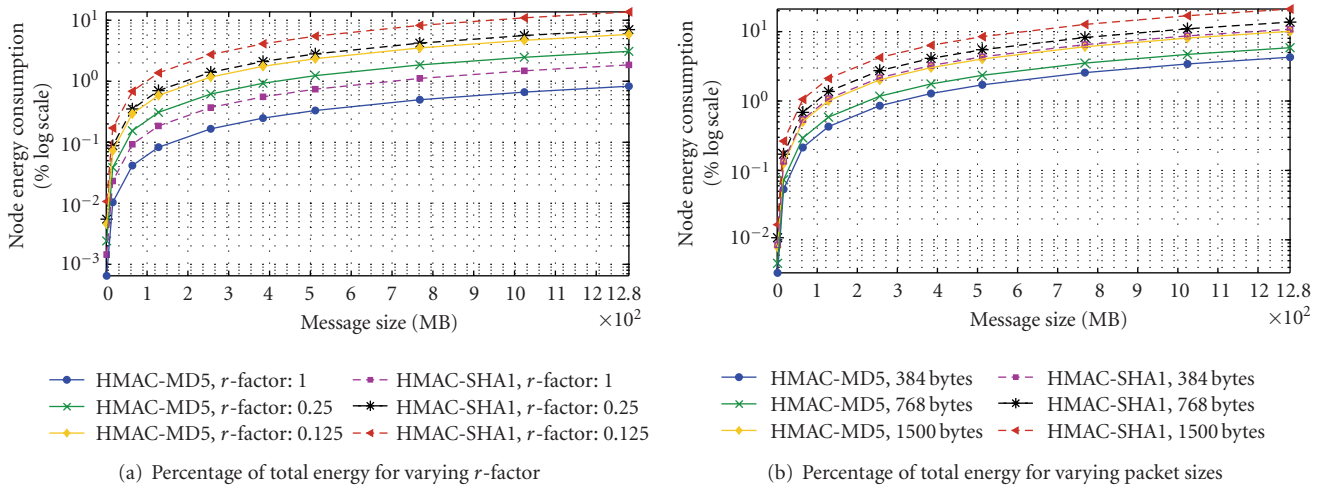


Figure 16: Total energy required for authenticating 500 MB data, ARM1176JZ(F).

Figure 14 compares the amount of energy that would be required to authenticate 500 MB data on the iPAQ handheld for different authentication algorithms. Clearly, authenticating 500 MB data without additional energy sources is not possible except for the proposed protocol.

Given the energy constraints above, authenticating 5 MB data on the handheld is more realistic—Figure 15(a) compares the percentage of the total node energy that is spent in authenticating 5 MB data. The graphs validate our claim that the energy consumption of the proposed protocol is significantly less in comparison to standard signature protocols, even efficient protocols like ECDSA. Moreover, the graphs show that for nodes with limited energy, the standard signature algorithms cannot be applied for authenticating every packet—even in the best case scenario, ECDSA signing algorithm consumes nearly 4% of the node energy for only 5 MB data. For the iPAQ handheld, the standard protocols can authenticate only a few mega bytes of data before they completely spend the available energy, as shown in Figure 15(b). Here also we assume that up to 50% maximum of the node energy may be spent in authenticating the data packets. The best scenario for the standard protocols is for RSA signature verification, where a node can authenticate nearly 543 MB of data if it is split into 1528 byte packets. The worst scenario is for RSA signature generation, where only 6 MB of data can be authenticated, for 668 byte packets. The extended TESLA protocol with HMAC-MD5 performs significantly better in comparison—being capable of authenticating 682 MB even with r-factor 0.125. The analyses above with iPAQ H3670 has the limitation that the processor power and battery capacity is on the lower end of the scale for mobile nodes that are available today. To illustrate the energy performance of the proposed protocol for a more current smart device, we have performed a simulation analysis of the energy consumption for different message sizes on the Apple iPhone processor, ARM1176JZ(F) [27]. The processor has an operating frequency of 620 Mhz and consumes 0.45 mW per cycle. We consider the node energy capacity to be equivalent to the iPhone battery capacity, 1400 mAh at 3.7 V [28]. Under the assumption that one HMAC operation is performed in one cycle of the processor, Figure 16 gives the total energy consumed for authenticating 500 MB data, for both HMAC-MD5 and HMAC-SHA1. As is expected, HMAC-SHA1, r-factor 0.125, consumes the most energy. However, even in the worst case, the 1541 joules of energy consumed represent 8.2% of the total battery capacity.

Figure 17 shows how much energy is consumed for authenticating varying message sizes. Figure 17(a) gives the values for the node energy as a percentage of the battery capacity for both HMAC-MD5 and HMAC-SHA1, when the r-factor is varied. Here the individual UDP payload is held constant at 768 bytes. Even the highest consumption—HMAC-SHA1, r-factor 0.125 authenticating 1280 MB data—is only 13.71% of the total battery capacity.



(a) Percentage of total energy for varying r-factor

(b) Percentage of total energy for varying packet sizes

Figure 17: Energy consumption of extended TESLA protocol on ARM1176JZ(F) for different message sizes.

When the packet size is varied, while keeping the r-factor constant (0.25), the energy consumption percentage is shown in Figure 17(b). The maximum consumption is again for HMAC-SHA1, for UDP payload size 1500 bytes, but it is still an acceptable 21.16% of the total energy. Even though these figures are only approximate, they demonstrate that the proposed source authentication protocol can be efficiently used for authentication in present-day smart devices.

It is to be noted that the energy consumption figures above are only for the computation costs of various protocol functions. Another primary source of energy consumption is message transmission. An evaluation of the computation and transmission costs based on the packet sizes, traded off against the packet error probability in transmission, may yield a packet size that is optimal in terms of overall energy consumption. Such analysis is beyond the scope of the current work.

9. Conclusion

In this paper, we have proposed a modified version of a new class of lightweight, symmetric-key certificates called TESLA certificate, and described a source authentication protocol for group communication in hybrid satellite/wireless networks that is based on the extended TESLA certificate. The extended TESLA certificate and the authentication protocol are ideally suited for wireless smart devices with limited energy availability. The certificate binds the identity of a sender device to the anchor element(s) of the device's MAC key chain(s). Binding the identity of a device to its key chain extends the lifetime of the device's certificate to multiple uses. Messages sent from the device are authenticated by MACs computed with keys from the chain. For the authentication protocol, we have used the satellite as the Certificate Authority to generate and distribute the extended TESLA certificates to all the nodes in the network. We have also used the satellite as the proxy node for the senders in disclosing the MAC keys to the receivers in the network. Furthermore, we have proposed a novel concept of probabilistic nonrepudiation that is based on having the satellite node as the proxy for key disclosure to the receivers. In terms of performance, due to the use of symmetric MAC functions, the proposed source authentication protocol expends much less processing power and node energy of the smart devices, in comparison to authentication protocols based on public key-based digital signatures. Through analysis and simulations, we have shown that the performance of the proposed authentication protocol is superior to using public key-based authentication mechanisms, for the metrics node energy and processing delay. We have also shown through security analysis that the source authentication protocol is secure against malicious adversaries.

There has been limited research in group authentication protocols that address the unique characteristics and requirements of wireless devices in hybrid networks such as the one we have described in this paper. We believe our proposed protocol makes a worthwhile contribution to address the paucity. Although we have described the source authentication protocol primarily in the context of hybrid wireless/satellite networks, with minor modifications the protocol can be made applicable to devices in generic wireless networks, provided that some centralized infrastructure is present for performing the functions of the CA and the proxy. Also, the nonrepudiation mechanism can be separately used with other symmetric MAC-based source authentication protocols for group communications, provided a trusted infrastructure is present for proxy key disclosure and to provide arbitration.

Acknowledgments

The authors would like to thank the anonymous reviewers whose valuable critique has contributed to improvements in this paper. The material presented in this paper is based upon work supported by National Aeronautics and Space Administration under award No. NCC8235, and by the

Defence Advanced Research Projects Agency (DARPA) award number 013641-001 to the University of California-Berkeley, under the MuSyC center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

References

[1] A. Roy-Chowdhury and J. S. Baras, "Improving network performance in hybrid wireless networks using a satellite overlay," in *Proceedings of the 13th Ka and Broadband Communications Conference*, Instituto Internazionale delle Comunicazioni (IIC), Turin, Italy, September 2007.

[2] National Institute of Standards and Technology (NIST), "Digital signature standard (DSS)," Tech. Rep. FIPS 186-3, NIST Information Technology Laboratory, Gaithersburg, Md, USA, March 2006.

[3] V. Gupta, S. Gupta, S. Chang, and D. Stebila, "Performance analysis of elliptic curve cryptography for SSL," in *Proceedings of the ACM Wireless Internet Security Workshop (WiSe '02)*, pp. 87–94, ACM, Atlanta, Ga, USA, September 2002.

[4] P. Prasithsangaree and P. Krishnamurthy, "On a framework for energy-efficient security protocols in wireless networks," *Computer Communications*, vol. 27, no. 17, pp. 1716–1729, 2004.

[5] S. Seys and B. Preneel, "Power consumption evaluation of efficient digital signature schemes for low power devices," in *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, (WiMob '05)*, vol. 1, pp. 79–86, IEEE, 2005.

[6] W. Freeman and E. Miller, "Experimental analysis of cryptographic overhead in performance-critical systems," in *Proceedings of the IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOT '99)*, pp. 348–357, College Park, Md, USA, October 1999.

[7] B. Kaliski, "Twirl and RSA key size," in *CryptoBytes Technical Newsletter*, RSA Laboratories, Bedford, Mass, USA, 2003.

[8] M. Bohge and W. Trappe, "TESLA certificates: an authentication tool for networks of compute-constrained devices," in *Proceedings of the 6th International Symposium on Wirless Personal Multimedia Communications (WPMC '03)*, Yokosuka, Japan, October 2003.

[9] A. Roy-Chowdhury and J. S. Baras, "A lightweight certificate-based source authentication protocol for group communication in hybrid wireless/satellite networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '08)*, pp. 1897–1902, IEEE, New Orleans, La, USA, December 2008.

[10] A. Roy-Chowdhury and J. S. Baras, "Probabilistic non-repudiation for source authentication with TESLA certificates in hybrid satellite/wireless networks and performance analysis of the authentication protocol," in *Proceedings of the 15th Ka and Broadband Communications Navigation and Earth Observation Conference*, Italian Space Agency, Cagliari, Italy, September 2009.

[11] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," in *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societie (INFOCOM '99)*, pp. 708–716, March 1999.

[12] R. Gennaro and P. Rohatgi, "How to sign digital signatures," in *Advances in Cryptology—CRYPTO*, pp. 180–197, Springer, Berlin, Germany, 1997.

[13] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," in *Proceedings of the 1998 International Conference on Network Protocols (ICNP '98)*, pp. 198–209, Austin, Tex, USA, October 1998.

[14] P. Rohatgi, "Compact and fast hybrid signature scheme for multicast packet authentication," in *Proceedings of the 6th ACM Conference on Computer and Communications Security (ACM CCS '99)*, pp. 93–100, ACM, Singapore, November 1999.

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

[16] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," in *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS '01)*, pp. 28–37, November 2001.

[17] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "TheTESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. 2, pp. 2–13, 2002.

[18] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of the Network and Distributed System Security Symposium (NDSS '01)*, 2001.

[19] M. Bohge and W. Trappe, "An authentication framework for hierarchical ad hoc sensor networks," in *Proceedings of the ACM Workshop on Wireless Security (WiSE '03)*, pp. 79–87, ACM, San Diego, Calif, USA, August 2003.

[20] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pp. 33–43, ACM, New York, NY, USA, 1989.

[21] "Secure Hash Standard," http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[22] D. Mills, RFC 1305—Network Time Protocol (Version 3) Specification, Implementation and Analysis, Internet Engineering Task Force (IETF), March 1992.

[23] J. Jonsson and B. Kaliski, Public-Key Cryptography Standards (PKCS) # 1: RSA Cryptography Specifications Version 2.1 (RFC 3447), Internet Engineering Task Force (IETF), February 2003.

[24] X. Ding, D. Mazzocchi, and G. Tsudik, "Equipping smart devices with public key signatures," *ACM Transactions on Internet Technology*, vol. 7, no. 1, article 3, 2007.

[25] "Compaq iPAQ Pocket PC H3600 series," http://h18002.www1.hp.com/products/quickspecs/10632div/10632div.HTML#QuickSpecs.

[26] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols," *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 128–143, 2006.

[27] "ARM1176JZ(F)-S processor," http://www.arm.com/products/CPUs/ARM1176.html.

[28] "iPod and iPhone battery and power specifications," http://www.ipodbatteryfaq.com/ipodbatteryandpower.html.