# Probabilistic Non-Repudiation for Source Authentication with TESLA Certificates in Hybrid Satellite/Wireless Networks and Performance Analysis of the Authentication Protocol

Ayan Roy-Chowdhury
Maryland Hybrid Networks Center, Institute for Systems Research, University of Maryland, College Park, MD 20742, USA. +1(301)4056561, ayan@umd.edu
John S. Baras
Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA. +1(301)4056606, baras@umd.edu

## Abstract

In this work, we describe a novel non-repudiation mechanism for an authentication protocol based on the extended TESLA certificate construct. With the non-repudiation mechanism, the authentication protocol is ideally suited for source authentication of low-powered nodes that participate in group communication in hybrid satellite/wireless networks.

Security is a necessary parameter in hybrid wireless networks (consisting of groups of terrestrial wireless nodes interconnected by a satellite overlay) if the communication between a pair of nodes, or amongst a group of nodes, is to be protected from unauthorized The focus of our research work is on user authentication and message integrity protocols, which are required to enable communications and ensure that messages between communicating nodes are correctly delivered. This is a non-trivial problem in group communication, where authentication has been traditionally done using asymmetric cryptographic techniques such as public key certificates and digital signatures. However, such asymmetric techniques can be very expensive in terms of processing power and node energy for wireless nodes in hybrid networks. As a solution to this problem, we have proposed a new class of lightweight, symmetric key certificates called extended TESLA certificates, and we have designed an energy-efficient source authentication protocol for group communication that utilizes the infrastructure present in hybrid satellite networks.

In our authentication protocol based on the extended TESLA certificate, we propose to add non-repudiation by taking advantage of the satellite infrastructure and the proposed mechanism of key disclosure by proxy. This is a major improvement over previous authentication algorithms based on TESLA, which do not provide for non-repudiation due to the symmetric nature of the underlying cryptographic primitive, Message Authentication Codes. In this paper, we describe the mechanism by which non-repudiation is achieved in our authentication protocol. The non-repudiation is probabilistic in nature, and we analyze the tradeoff between the degree of non-repudiation and the overhead due to the additional information necessary for non-repudiation. Furthermore, through simulations we compare the performance between the basic authentication protocol and authentication with non-repudiation, and also compare the latter to public key-based schemes.

## 1. Introduction

Large wireless networks have the ability to provide rapid connectivity in disaster areas or military battlefields, or to inter-connect users in far-flung geographical locations. 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 clusters of terrestrial wireless nodes with dual satellite connectivity providing alternate high bandwidth and robust forwarding paths through satellite links.

Security is a necessary parameter in hybrid wireless networks to protect the communication amongst user nodes from unauthorized access or unauthorized modifications. 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 nodes are properly authenticated to *enable* communication. Authentication in group communication is traditionally done by digital

signatures [2], based on public-key cryptography. However, generation and verification of digital signatures for messages are expensive in CPU cycles and therefore energy expenditure [3]–[5]. In wireless networks where many nodes have limited processor power, storage capacity and available energy, performing digital signature generation and verification frequently can prevent the CPU from other functions and drain the battery quickly. Therefore in hybrid wireless networks it is preferable to use authentication protocols that are based on symmetric cryptographic primitives like Message Authentication Code or MAC (for example, HMAC [6]) - which expend less node energy. However, designing authentication protocols for group communication using symmetric cryptography is a significant challenge. The primary difficulty in designing a scalable scheme is how to create the asymmetry efficiently such that receivers can authenticate the messages without heavy expenditure of system resources.

We have proposed to achieve authentication in hybrid satellite/wireless networks using a new class of certificates called *TESLA Certificate*. The TESLA certificate concept was originally proposed in [7], and it was used to build a basic authentication scheme in [8]. In [9], we have extended the TESLA certificate design and used the extended construct to propose an authentication protocol for hybrid networks, taking advantage of the presence of the satellite overlay network. In our proposed authentication 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 in group communication.

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. This is considered an essential element for building a comprehensive authentication and message integrity protocol. In comparison, the TESLA authentication algorithm [7] or the prior TESLA certificate proposal [8] do not provide non-repudiation. The symmetric nature of the basic cryptographic primitive used here - MACs - does not allow for non-repudiation. 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 non-repudiation is a major drawback of the TESLA certificate protocol compared to digital signatures based on public keys. In our extended TESLA certificate algorithm, we add a probabilistic non-repudiation mechanism by taking advantage of the satellite infrastructure and the proposed mechanism of key disclosure by proxy. In this paper, we describe the method by which probabilistic non-repudiation is added to the TESLA authentication algorithm. In addition, we analyze the tradeoff between the degree of non-repudiation and the overhead due to the additional information necessary for non-repudiation. Through simulations we also compare the performance between the base authentication protocol and authentication with non-repudiation, and also compare the latter to public key-based schemes.

The rest of the paper is organized as follows. In sections 2 and 3, we briefly review the TESLA algorithm and the original TESLA certificate protocol, respectively. We describe our proposed source authentication protocol using TESLA certificates in section 4. The non-repudiation mechanism for the authentication protocol is described in section 5. An analysis of the protocol performance with and without non-repudiation is given in section 6. We conclude with a brief discussion in section 7.

## 2. Review of TESLA Authentication Protocol

The TESLA protocol [7] 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.

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 corresponds to the number of intervals in which the source transmits. The source computes the MAC key for each interval by applying a second public one-way function $F_2$ to each element of the hash chain. The algorithm is illustrated in fig. 1.

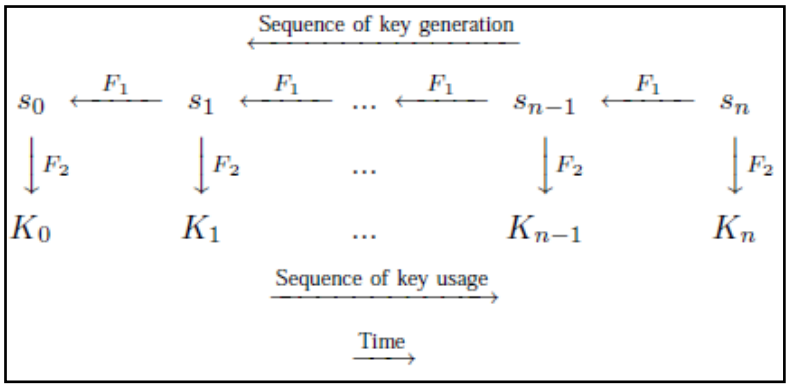


Fig. 1 TESLA key generation

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. The sender bootstraps the hash chain by broadcasting to all the receivers the anchor element of the chain, $s_0$. For each packet generated in time slot $i$, the sender uses the authentication key $K_i$ to compute a MAC on the packet. The sender discloses $K_i$ at a later time instant by broadcasting the corresponding key seed $s_i$. Upon receiving $s_i$, each receiver first verifies the authenticity of $s_i$ by checking $s_i \xrightarrow{F1} s_{i-1} \xrightarrow{F1} \ldots \xrightarrow{F1} s_0$. If $s_i$ verifies correctly, each receiver can compute $K_i : s_i \xrightarrow{F2} K_i$ and subsequently use $K_i$ to verify the MAC on the packets received during 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. This is the principle of delayed disclosure of keys.

## 3. Review of TESLA Certificate Protocol

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 ${}_aK_{Bn}$ for $B$ to use to compute the MAC on its messages in that interval. The CA creates a certificate $Cert_{CAn}(B)$ to bind ${}_aK_{Bn}$ to $B$ for interval $n$. The CA uses its TESLA key $tK_{CAn}$ to encrypt ${}_aK_{Bn}$ in the certificate, and uses the same key to compute a MAC on the certificate: $Cert_{CAn}(B) = (IDB,\{{}_aK_{Bn}\}tK_{Can}, n+d, MAC_{tKCAn}(..))$. ${}_aK_{Bn}$ is known only to the CA and $B$ during period $n$, while $tK_{CAn}$ is known only to the CA. $n+d$ indicates the time at which the CA will disclose $tK_{CAn}$ to the nodes, that is, it is the expiration time of the certificate. The CA sends $Cert_{CAn}(B)$ to $B$ along with ${}_aK_{Bn}$, which is encrypted with key $K_{CA,B}$ that is shared between the CA and $B$.

In the time interval $<n, n+d_i>$, $D$ 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: $B \rightarrow D$: $(Cert_{CAn}(B), MAC_aK_{Bn}(request))$. When $D$ receives the authentication message, it checks the timestamp of $Cert_{CAn}(B)$ to make sure it has arrived before time $n+d$. At time $n+d$, the CA discloses $tK_{CAn}$. Upon receiving the key, $D$ verifies $Cert_{CAn}(B)$ by checking the MAC in the certificate using $tK_{CAn}$. If the MAC verifies correctly, $D$ obtains ${}_aK_{Bn}$ from the certificate by decrypting with $tK_{CAn}$. Subsequently, $D$ checks $MAC_aK_{Bn}(request)$ to verify the authenticity of $B$.

A TESLA certificate, as envisioned in [8], 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.

## 4. Extension to TESLA Certificate and Authentication in Hybrid Networks

We extend the lifetime of the TESLA certificate from single use to multiple uses by combining key chains with the certificate. We use this extended TESLA certificate in the authentication protocol proposed in [9] and disclose TESLA keys used by the source node by allowing the satellite in the hybrid network to broadcast the sender's MAC keys to the receivers – the satellite therefore acts as a proxy for the sender. We also use the satellite as the CA to generate the TESLA certificates for the sender nodes. The authentication protocol is described in brief as follows.

### 4.1 Setup: Key Chain Generation by CA and Source Node

During the initial setup, before any messages are transmitted, the CA and all sources generate the key chains that each use for message authentication.

The CA uses a key chain $\{tK_{CA,i}\}$ (where $i = 1, ..,N$) to authenticate the TESLA certificates that it creates. It starts with a random seed $s_{CA,N}$ and applies one-way function $F_1$ to $s_{CA,N}$ to form a TESLA chain in a manner similar to that described in section 2. Subsequently the CA applies function $F_2$ to each element $s_{CA,i}$ of the TESLA chain obtain key $tK_{CA,i}$. $s_{CA,0}$ 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. $s_{CA,0}$ is broadcast to the network in time $t < t_0$: $CA \rightarrow network: (s_{CA,0}, SIGN_{-KCA}(..))$ where $-K_{CA}$ is the private key of the CA's public-private key pair.

Similar to the CA, 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. $A$ subsequently applies $F_2$ to each key $s_{A,i}$ of the chain and obtains $s'_{A,i}$. At time $t < t_0$, $A$ sends $s_{A,n}$ and $n$ 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 $K_{CA,A}$ between $A$ and the CA. The CA can obtain all the elements of $A$'s TESLA key chain from $s_{A,n}$ and $n$.

On successful verification of $A$'s identity, the CA generates the TESLA certificate for $A$. The key $s_{A,0}$ is included in the certificate as the anchor element of $A$'s key chain. It is encrypted using key $tK_{CA,1}$ from the CA's key chain. The certificate also includes the identity of the source node $A$ and the time $t_0+d$ up to which the certificate is valid, i.e., 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 also contains a MAC for authentication, computed on the previous elements using $tK_{CA,1}$. For added security, the certificate may also be signed by CA's digital signature: $Cert_{CA}(A) = (ID_A,\{s_{A,0}\}tK_{CA,1},t_0+d,MAC_{tKCA,1}(..),SIGN_{-KCA}(..))$. Here $d$ is the key disclosure delay for the CA TESLA signature key, and $tK_{CA,1}$ is the CA MAC key for the time period $<t_0, t_0+d_i>$.

### 4.2 Message Transmission from Source to Receiver

$A$ sends messages $M_0..M_i$ to $B$ and $C$ starting in the time interval $<t_0, t_0+d_i>$. A computes a MAC over the message $m_0$ using $s'_{A,0}$ and includes its TESLA certificate $Cert_{CA}(A)$ with the message: $A \rightarrow \{B,C\}:\{M_0|M_0 : m_0,MAC_{s'A,0}(m_0),Cert_{CA}(A)\}$. Each of $B$ and $C$ checks the *freshness* of the certificate by checking the timestamp of $Cert_{CA}(A)$ to make sure it has arrived within the period $<t,t_0+d_i>$. The receivers also check that $s'_{A,0}$ is not publicly known, i.e., $MAC_{s'A,0}(m_0)$ 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.

### 4.3 Message Authentication at Receiver

At time $t_1 = t_0+d$, the CA broadcasts the key $s_{CA,1}$: $CA \rightarrow network : (<t_0, t_0+d_i>, s_{CA,1}, SIGN_{-KCA}(..))$. Receiver $B$ or $C$ can check the authenticity of $s_{CA,1}$ by verifying $s_{CA,1}$ against the anchor element $s_{CA,0}$: $s_{CA,1} \overset{F}{1}\rightarrow s_{CA,0}$. $B$ or $C$ can alternatively verify $s_{CA,1}$ from the digital signature using CA's public key $+K_{CA}$. If verification is successful, each receiver derives $tK_{CA,1}$ from $s_{CA,1}$ using $F_2$ and uses $tK_{CA,1}$ to verify the MAC on $Cert_{CA}(A)$. If the MAC is correct, each receiver obtains $s_{A,0}$ from $Cert_{CA}(A)$ by decrypting with $tK_{CA,1}$. $B$ or $C$ obtains $s'_{A,0}$ from $s_{A,0}$, then checks $MAC_{s'A,0}(m_0)$ using $s'_{A,0}$ and accepts $m_0$ if the MAC verifies correctly. Each receiver saves $Cert_{CA}(A)$ 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 $A$ in subsequent time intervals are authenticated using the corresponding key of $A$'s key chain. $A$ 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 $<t_i, t_i+i>$, message $M_i$ from $A$ to $B$ would look like: $A \rightarrow B: \{M_i \mid M_i: m_i,MAC_{s'A,i}(m_i)\}$. At time $t_i+d$, the CA broadcasts $s_{A,i}$ to the network. 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: $s_{A,i} \overset{F}{1}\rightarrow s_{A,i-1} \overset{F}{1}\rightarrow ... \overset{F}{1}\rightarrow s_{A,0}$ where $s_{A,0}$ has already been verified from $Cert_{CA}(A)$.

## 5. Non-repudiation with TESLA Certificate Authentication in Hybrid Networks

In the extended TESLA certificate authentication protocol described in section 4, we propose to add non-repudiation by taking advantage of the satellite infrastructure and the proposed mechanism of MAC key disclosure by proxy. This is achieved as described in the following section.

The authentication protocol in section 4 is described using one MAC per message. For allowing non-repudiation, we propose that the source authenticates each message by *two or more* MACs, computed using keys from two or more key chains, respectively. The anchor element of the root key of each chain is shared between the source and the CA (the satellite) as described in 4.2. 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 4.2.6. 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 non-repudiation 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., non-repudiation), the receiver can simply send a non-repudiation 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 operate 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 non-repudiation 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 non-conforming source who includes only one correctly computed MAC with its message in order to avoid non-repudiation, 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 drops to 0.25, and so on. There is hence a trade-off between the strength of the non-repudiation algorithm and the security overhead per message in terms of number of MACs involved. There is also the processing overhead at the source since it 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 non-repudiation mechanism will be successful for the request from one receiver. For any subsequent request from other receivers for that particular message, non-repudiation will fail since both MAC keys are now known to the receivers. The number of successful non-repudiation 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 non-repudiation. 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.

## 6. Performance Analysis

As part of simulations conducted for quantifying the performance of the proposed authentication algorithm, we have analyzed the overhead of allowing non-repudiation. The simulations show that 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.

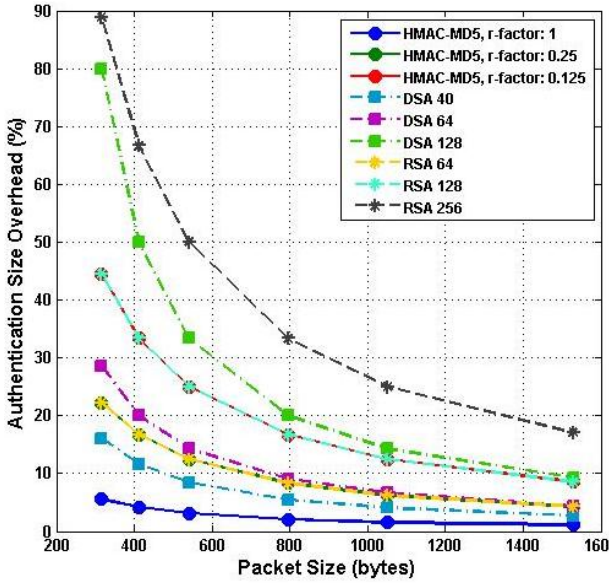


Fig. 2 Comparison of percentage byte overhead between TESLA with HMAC-MD5 and DSA, RSA

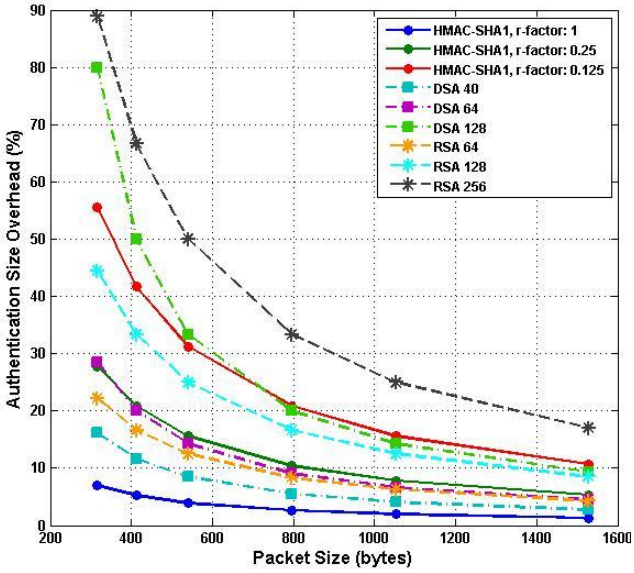


Fig. 3 Comparison of percentage byte overhead between TESLA with HMAC-SHA1 and DSA, RSA

A comparison of the size overhead incurred for authenticating 500MB data using our proposed protocol or with public-key algorithms DSA or RSA, is shown in figures 2 and 3. Figure 2 compares extended TESLA with HMAC-MD5 (size of one MAC = 128 bits) against DSA and RSA, while figure 3 compares extended TESLA with HMAC-SHA1 (size of one MAC = 160 bits) against DSA and RSA. 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.

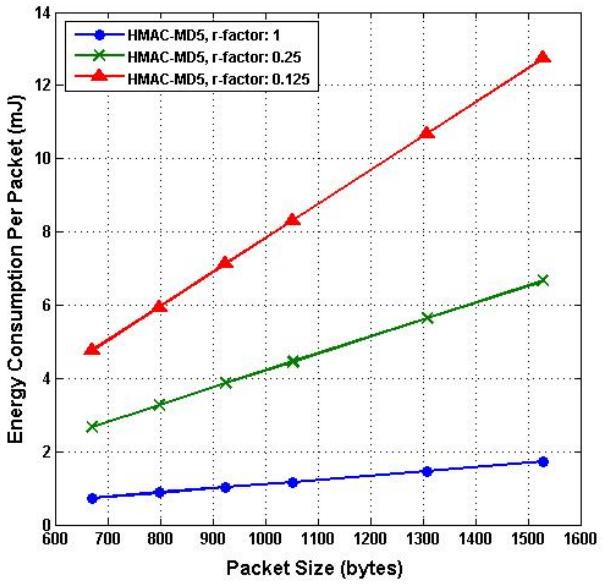


Fig. 4 Energy consumption per packet for HMAC-MD5 authentication

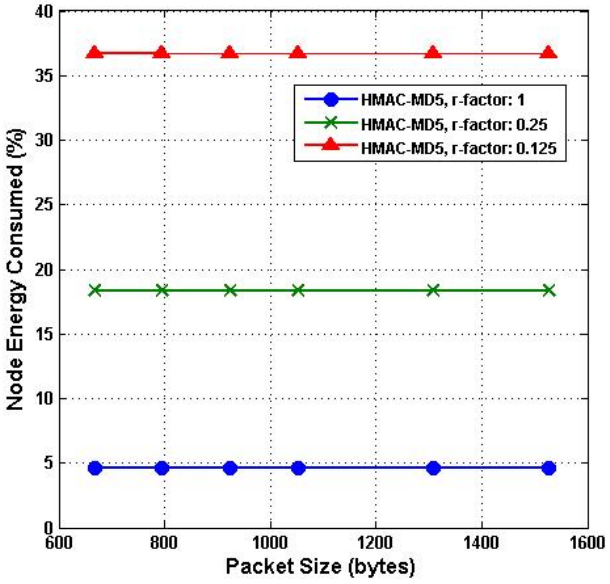


Fig. 5 Percentage node energy consumed for HMAC-MD5 authentication of 500MB data,

For extended TESLA, we consider three cases based on the degree of non-repudiation present - for each packet that is authentication, r-factor is either 1 (one MAC), or 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 (for modulus N=512,1024 and 2048 respectively). 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 non-repudiation, 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 non-repudiation scheme. However, even then the overhead for our protocol is significantly less than the overhead due to strong RSA (256 byte signature) or DSA (128 byte signature)-based security.

Another parameter where non-repudiation has a measureable impact is in the amount of node energy required for processing the authentication data. We simulate the energy consumption for a Compaq iPAQ H3670 handheld computer based on the energy expenditure measures of different crypto-graphic operations given in [11]. Figure 4 shows that the energy consumption for authenticating each packet ranges between 0.7238mJ and 12.73mJ, with larger packets and higher r-factors consuming more energy. Figure 5 shows the total energy consumed for authenticating 500MB 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 500MB data cannot be authenticated without recharging. 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 are indicative of the energy expense of the receivers.

Despite the higher energy requirement for non-repudiation, the TESLA authentication protocol performs significantly better than standard public-key algorithms like RSA and DSA even when non-repudiation is allowed. This is illustrated by figure 6, which compares the amount of data that can be authenticated using 50% of the node energy. As the figure illustrates, the standard protocols can authenticate only a few mega-bytes of data before they completely spend the available energy. The best scenario for the standard protocols is for RSA signature verification, where a node can authenticate nearly 543MB of data if it is split into 1528 byte packets. The worst scenario is for RSA signature generation, where only 6MB 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 682MB even with r-factor 0.125.

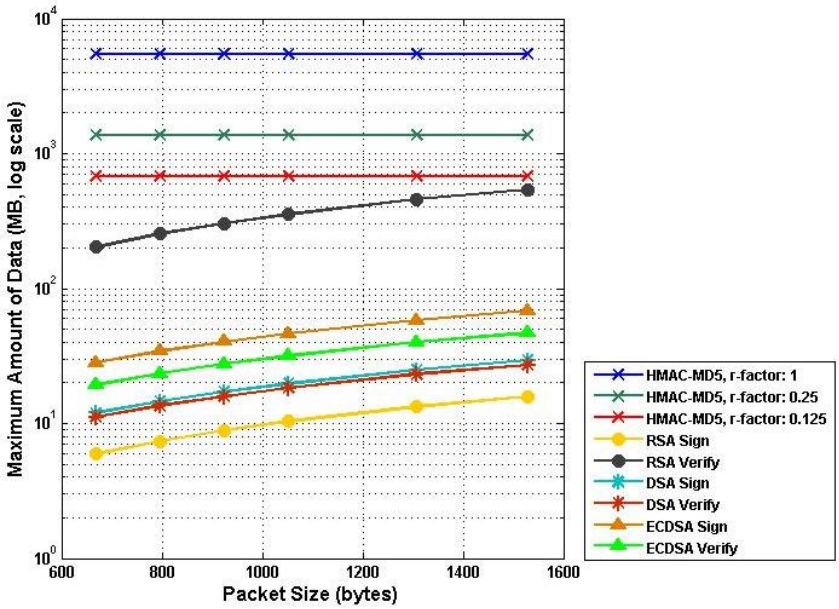


Fig. 6 Comparison of maximum data processable.

## 7. Conclusion

In this paper we have described a novel concept of probabilistic non-repudiation that is used as part of the TESLA certificate multicast authentication protocol in hybrid satellite/wireless networks. The non-repudiation mechanism makes use of having the satellite node act as the source node's proxy for key disclosure to the receivers. This allows the satellite node to be used as an arbitrator when the requirement for non-repudiation arises. Through simulations we have analyzed the trade-off of the

non-repudiation mechanism. We have shown that the non-repudiation has a higher byte overhead and higher energy consumption compared to authentication without repudiation. However, the simulations also show that the authentication protocol with non-repudiation still performs significantly better than comparable public cryptography-based authentication algorithms.

Even though we have described the non-repudiation mechanism as a part of our proposed authentication protocol, the method can also be used with other symmetric-MAC based multicast authentication protocols, provided a trusted infrastructure is present for proxy key disclosure and to provide arbitration.

## Acknowledgement

The material presented here is based upon work supported by National Aeronautics and Space Administration under award No. NCC8235. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

## References

1 A. Roy-Chowdhury and J. S. Baras, “Improving network performance in hybrid wireless networks using a satellite overlay,” in Proc. 13th Ka and Broadband Communications Conference. Turin, Italy: Instituto Internazionale delle Comunicazioni (IIC), September 24-26 2007.

2 N.I.S.T., “Digital signature standard (DSS),” May 19 1994.

3 P. Prasithsangaree and P. Krishnamurthy, “On a framework for energy efficient security protocols in wireless networks,” *Elsevier Computer Communications*, vol. 27, pp. 1716–1729, 2004.

4 S. Seys and B. Preneel, “Power consumption evaluation of efficient digital signature schemes for low power devices,” in *Proc. 2005 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (IEEE WiMOb 2005)*, vol. 1. IEEE, 2005, pp. 79–86.

5 W. Freeman and E. Miller, “An experimental analysis of cryptographic overhead in performance-critical systems,” in *Proc. 7th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOT’99)*. College Park, MD, USA: IEEE, October 1999, pp. 348–357. [Online]. Available:citeseer.ist.psu.edu/freeman99experimental.html

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

7 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.

8 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.

9 A. Roy-Chowdhury and J. Baras, “A lightweight certificate-based source authentication protocol for group communications in hybrid wireless/satellite networks,” in Proc. IEEE Global Communications Conference (Globecom) 2008. New Orleans, Louisiana, USA: IEEE, November 30 - December 4 2008.

10 A. Roy-Chowdhury, “Improving network performance, security and robustness in hybrid wireless networks using a satellite overlay,” Ph.D. dissertation, University of Maryland, College Park, Maryland, USA, September 2008.

11 Compaq iPAQ Pocket PC H3600 series. http://h18002.www1.hp.com/products/quickspecs/10632 div/10632 div.HTML#QuickSpecs.