Automated key exchange protocol evaluation in delay tolerant networks

Sofia Anna Menesidou, Dimitrios Vardalis

Information Security and Incident Response Unit
Department of Electrical and Computer Engineering
Democritus University of Thrace
University Campus, Xanthi 67100, Greece
{smenesid,dvarali}@ee.duth.gr
http://isir.ee.duth.gr/

Vasilios Katos

Cyber Security Unit
Department of Computing and Informatics
Bournemouth University
Poole House, Fern Barrow, BH12 5BB, UK
vkatos@bournemouth.ac.uk
http://bucsu.bournemouth.ac.uk/

Abstract

Cryptographic key exchange is considered to be a challenging problem in Delay Tolerant Networks (DTNs) operating in deep space environments. The difficulties and challenges are attributed to the peculiarities and constraints of the harsh communication conditions DTNs typically operate in, rather than the actual features of the underlying key management cryptographic protocols and solutions. In this paper we propose a framework for evaluation of key exchange protocols in a DTN setting. Our contribution is twofold as the proposed framework can be used as a decision making tool for automated evaluation of various communication scenarios with regards to routing decisions and as part of a method for protocol evaluation in DTNs.

Keywords: delay tolerant networks, protocol evaluation, opportunistic key management, protocol injection, ns-2

1. Introduction

Delay or disruption tolerant networks (DTNs) have increasingly become popular due to certain advantages over traditional protocols such as TCP/IP. DTNs are store-and-forward networks and can be applied in various connectivity-"challenged" networks such as deep space networks, under-water networks, vehicular networks, sensor networks, mobile ad-hoc networks and so forth, where bandwidth is limited and an end-to-end path from source to destination is not always available. Although DTNs by nature may support high availability (which in DTN terminology is referred to as reliability), they are not short of security issues.

Over the past few years, cyber attacks have grown constantly emphasizing the need and importance of information security within a network. Security is one of the major issues in deep space networks not only for military, governmental, and commercial missions but also for scientific projects. Cryptographic key exchange and cryptographic key management in general is considered to be a challenging problem in DTN environments. Although there is a wealth of key management protocols in the literature, there is no practical mechanism to evaluate which protocol is more efficient in an environment with limited connectivity and bandwidth. Throughout the literature the emphasis and priorities in cryptographic protocol analysis were placed on the security goals rather than the quality of service and the applicability and feasibility of such protocols. Albeit the evolution, maturity and existence of rigorous and formal tools for assessing the security properties of the plethora of security protocols, many of them may be impractical in a DTN environment. As such, the purpose of this paper is to propose a framework for evaluating key agreement protocols in terms of delay and bandwidth constraints. From our simulation results we will also show the impact of the missing credentials (e.g. certificate or session key) on the end-to-end delay of space DTNs. The novelty of this work lies on the area of cryptographic key management in DTNs which is an open and challenging task.

The remainder of this paper is structured as follows. Section 2 provides the literature review and the current state of the art for key management in DTNs. In Section 3 a framework to evaluate cryptographic protocols in such networks is proposed and in Section 4 simulation results are presented. In Section 5 a discussion on opportunistic key management as an idea to minimize end-to-end delay is introduced. Finally, we conclude the paper with Section 6 where open issues, research challenges and future work directions are summarized.

2. Related work and current state of the art

The literature has a relatively long lasting and mature domain of key management. A significant amount of work is published on key management protocols, most of which have been extensively analyzed, with well recognized security properties and acknowledged weaknesses. However when it comes to integrating such protocols in a DTN infrastructure, it seems that the requirements and constraints of a DTN environment render such protocols unsuitable. As such, the prohibiting factors relate to the practical communication, performance and efficiency aspects rather than the security capabilities of the key management protocols. Most of the work done until now is based on the assumption of shared key material [1]. In [2] the author states a series of requirements for key management in DTNs without proposing a solution. Up to date a limited number of approaches for key management in DTN environments can be found in the literature, yet no method for automated practical evaluation of key establishment has been proposed. In fact in RFC6257 [3] and in the Internet draft [4] key management is recognized as a difficult topic and the authors explicitly state that such exclusion is a result of an informed decision. Last but not least, the relatively new Internet draft proposes and outlines a design for security key management in DTNs [5]. Specifically, the core requirements and design criteria for DTN security key management are described.

2.1. Key management in DTNs

Identity-Based Cryptography (IBC) has been examined as a potential solution for security within DTNs. IBC is a cryptographic method that enables message encryption and signature verification using the public identifier of an entity. The authors in [6] propose an Hierarchical Identity Based Cryptography (HIBC) that provides efficient and practical solutions to secure channels, mutual authentication and revocations in DTNs. In addition, the authors in [7] evaluate IBC cryptography in the context of a DTN. In [8] an anonymous authentication protocol is presented and a secure communication solution based on the non-interactive Sakai-Ohgishi-Kasahara (SOK) key agreement scheme is proposed. This scheme is based on Boneh-Franklin hierarchical identity based encryption (IBC) and signature schemes. However, such IBC solutions appeared to superficially solve the problem [2]. Specifically, IBC solutions are undesirable due to intractability of some problems such as Private Key Generation (PKG) parameter distribution, private key revocation, identity name space management and so forth [9].

In [10] a group-oriented security solution for DTN that provides access control and secure group communications is proposed. The authors suggest a centralized group key management mechanism of the logical key hierarchy (LKH). Group key management in DTN has been studied in [11]. The proposed protocol is based on the Chinese Remainder Theorem. The key lifetime concept is also introduced in order to alleviate the forward security problem in many-to-many DTN communication scenarios. Another work on group key management is found in [12] where an automatic group key management scheme based on one-encryption-key multi-decryption-key (OMPK) key protocol for DTNs is proposed. In this work the authors also prove forward and backward security of their protocol.

The author in [13] presents a key distribution scheme for infrastructureless networks, which is based on the Bundle Protocol (BP) and more specifically on the Bundle Security Protocol (BSP). With this dynamic and non-interactive scheme, cryptographic keys for all the BSP mechanisms can be established. The derived keys will be used by the BSP supported algorithms such as HMAC-SHA1 for authentication, RSA for digital signatures, and AES for encryption. However, this work is also based on IBC which has proved to be impractical for DTNs. The authors in [14] propose a dynamic virtual digraph (DVD) model for public key distribution. They heuristically define the DVD model by extending traditional graph theory. A public key distribution for pocket DTN based on two-channel cryptography is also presented. The authors in [15] propose a one-pass key establishment protocol based on an adoption of the Horsters-Michels-Petersen (HMP) protocol. In their method they inject protocol messages in the payload of the BP as part of the message. In addition, an encryption decision making workflow diagram of a custodian node is developed.

More recently, the work in [16] focuses on the problem of key management in the framework of content-based forwarding and opportunistic networks. A specific key management scheme that enables the bootstrapping of local topology is also proposed. In [17] the authors propose an authentication and key agreement protocol with anonymity based on combined public keys for DTNs. In their solution an on-line third trusted party is not required. In [9] the authors propose a non-interactive key establishment scheme for BSP. Their work focuses on space DTNs. They utilize a time-evolving mode based on the periodic and predetermined behavior patterns of space DTNs. From the model, they can schedule when and to whom a node should send his public key.

2.2. Key establishment evaluation

Selecting a suitable key management architecture from a large number of candidate protocols to satisfy a variety of communication scenarios is a challenging task making an automated evaluation method highly desirable. Authors in [18] proposed an evaluation index for a number of existing key management schemes using qualitative analysis. However, such proposals have limited value unless they take node replication attacks and robustness into consideration [19]. The authors in [19] employ Analytic Hierarchy Process (AHP) in order to select a suitable trustworthy key management scheme for wireless sensor networks

(WSN). They also state that the selection of a suitable key management scheme from a large number of existing schemes is not an easy issue. Six performance criteria are considered for the selection: scalability, key connectivity, resilience, storage overhead, processing overhead and communication overhead. All the aforementioned studies for protocol evaluation are based either on qualitative analysis or on the study of possible attacks. In DTNs and specifically in deep space environments where bandwidth is a scarce resource a framework for automated protocol evaluation is necessary. This framework for protocol evaluation can apply in various DTN networks and scenarios. To the best of our knowledge this is the first framework which takes into consideration node credentials (eg. access to session keys or certificates) and DTN network topology and evaluates the protocol based on the overhead imposed by the exchanged messages.

3. The Framework

The basic idea of our framework is to evaluate protocols and specifically key exchange protocols automatically. A protocol parser has been created for analyzing both cryptographic and environment delays in order to export a ns-2 script. The main focus of this study is to examine end-to-end delay considering various cryptographic delays. More specific, cryptographic message length, number of passes and network topology are analysed to produce end-to-end delay. The protocol parser has been implemented in python and can be used as a decision making tool through the evaluation of alternative communication paths and scenarios. As an input we use three representation languages, the protocol language (protocol messages), the node language (node credentials) and the path language (end-to-end paths with bandwidth information). From the protocol and node language we can extract the cryptographic delays while from the path language we can extract the transmission delay. We also made the assumption that the propagation and processing delay are constants with a fixed value (typical values for deep-space networks) as an input. However, the processing delay is negligible compared to transmission delay in such networks. In addition, in this work, inter-meeting time between nodes is not considered, due to a priori-known and predetermined connection time in deep-space DTNs. The aforementioned procedure is depicted in Fig.1. The parser also checks whether the protocol can be completed by the participating nodes prior to exporting the ns-2 script. The checks are performed for each of the participating nodes and establish whether the underlying node is capable of performing the respective protocol step - that is, whether it has the required knowledge of the protocol parameters. In the case of failing to pass the check, the parser recommends injection of further protocols to generate or obtain the missing parameters. Due to the lack of regular and timely communication in DTNs, security credentials could be missing or expired. As such, timely access to security credentials may not always be possible. In addition, due to the lack of connectivity, nodes may not have the opportunity to establish session keys. In cases of limited or no opportunities to obtain security credentials in advance, nodes can retrieve them afterwards resulting to a considerable increase in end-to-end delays as yielded by the parser, due to the message injection process (see Scenario 4). Naturally, such protocol injection could add a considerable amount of delay or overhead to the communication, which will be recorded by the ns-2 simulation.

Protocol Language. In the protocol language we use a symbolic model and describe the protocol by constructing a transcript of the exchanged messages. The usage of a symbolic model is suitable for automation [20]. Table 2 summarizes all the supported functions and a usage example. All the functions calculate and return the length (in bytes) of the cryptographic operation. To create the desirable message we can combine the supported functions in a row or even as an input for another function. The syntax requires that source and destination nodes are specified on the left side of the message. For instance, a simple nonce message between nodes A and B is represented as “A→B:nonce(4)”, where 4 is a typical length in bytes of a nonce message.

Node Language. In the node language we map the appropriate cryptographic parameters to nodes, namely symmetric key, private key, public key and certificate. In Table 3 we summarize the usage of the node language. For example if both nodes have the symmetric key K we can represented as "A:symkey=K" and "B:symkey=K".

Path Language. Last but not least we have the path language, where routing end-to-end paths are inserted. The path must correspond to the following pattern:

```
Source Node[Downlink,Uplink]Middle Node[Downlink,Uplink]Destination Node.
```

A usage example could be A[0.1Mb,0.0002Mb]B, where A is the source node, B is the destination, the downlink bandwidth (from A to B) is 0.1 Mbps and the uplink bandwidth (from B to A) is 0.0002 Mbps.

For the implementation of the aforementioned parsing languages we use pyparsing [21], which is python library for creating grammars. Apart from the input languages the protocol parser contains a library with a variety of key agreement protocols.

One other characteristic of the protocol parser is message or protocol injection that may be required as mentioned earlier. A usage example of the injection could be the following scenario. Assume that node A (security-source) wishes to exchange encrypted information with node B (security-destination) by using the AES algorithm. However, only node B has access to the symmetric key. The parser will automatically inject the appropriate messages between node A and a trusted node who serve as a Key Distribution Center (KDC), in order to have access to the appropriate credential. The exported tcl script will contain all the necessary communication for a successful scenario completion.

4. Simulation Results

Following the execution of a specific scenario by the parser, the exported tcl scripts can be fed to ns-2. More specifically, the simulations can be run on the DTN agent for ns-2 [22]. Modifications have been made in order to support two-pass communication and interactive communication protocols.

For the simulations we used TCP as the convergence layer, with a propagation delay of 10 min (which is a typical value for a deep space DTN scenario [23]) and a maximum bundle fragmentation size of 30 KB. We assume the following deep-space scenarios where all the available paths can accurately be predicted. The first two scenarios presented are trivial, and are used for creating a benchmark and for drawing conclusions on the delay overheads due to lack of access to security credentials and generally due to the number of protocol passes in deep-space networks. The rest of the scenarios are based on real deep-space scenarios, where data needed to be sent to ground earth station [23].

4.1. Scenario 1

In this scenario node A (security-source) wishes to establish a session key with node B (security-destination) by using an adoption of HMP protocol [15]. HMP is a one-pass key agreement protocol. To adopt HMP, protocol messages are injected in the payload of the Bundle Protocol (BP). We selected this protocol for simulation due to low communication cost by design. The downlink between A and B is 0.05 Mbps and the uplink between B and A is 0.0001 Mbps. We will compare two alternatives of the protocol: i) both nodes have all the necessary credentials and ii) the sender (node A) does not have access to receiver's (node B) public key.

In the first case only one protocol message (one-pass) is needed for a successful scenario completion. However in the second scenario two more messages must be exchanged prior to the protocol message: one message that asks for the public key from the Certificate Authority (CA) and one reply message from the CA to node A that contains the public key. Fig. 2 shows the first case, where both nodes have access to public key and the second case after the message injection.

From the simulation results (see Table 1), we note the significance of a node to have access to all the appropriate credentials. As we can see end-to-end delay is almost three times greater in the scenario where key establishment is needed. One approach for addressing delays due to the unavailability of credentials is the idea of opportunistic key exchange that is introduced in the next section.

4.2. Scenario 2

In this scenario, node A (security-source) wishes to establish a session key with node B (security-destination) and both nodes have access to all the necessary credentials. The proposed framework is used to compare a selection of one-pass key authenticated key agreement protocols. More specifically, we evaluated some of the existing key exchange schemes the Key Exchange Algorithm (KEA), the Unified Model (UM), the Menezes-Qu-Vanstone (MQV), the Station-to-Station (STS) and the adapted HMP protocol. The first three algorithms have the same exchanged messages (two-pass) and only the key calculation in the node changes. STS is a three-pass protocol, whereas the HMP based protocol is one-pass protocol.

From the protocol comparison (see Table 1) we can see the difference amongst the one-pass, two-pass and three-pass protocols. From the results it can be seen that one-pass key exchange protocols can be highly preferred in deep space scenarios due to their performance.

4.3. Scenario 3

In this scenario, we assume rover A (security-source) wants to send 1 KB of sensitive data to the ground station E (security-destination). There are two available paths: i) source A to satellite D to destination E and ii) source A to satellite B to satellite C to destination E. In the first path the middle node D is untrusted and confidentiality is a security requirement that must be fulfilled, whereas in the second path all nodes are trusted. In addition source node A (rover) does not have the session key and he must require it from a neighbor node which serves as a KDC.

The question is to identify which case is more efficient in terms of delay. Fig. 3 depicts the topology with all the possible routes from rover to ground station. In Table 4 we can see the input parameters of the protocol parser for both cases.

From the simulation results (see Table 1) we can conclude that it is more desirable to select the longer path because node A does not have the session key K. If the session key had been retrieved from KDC before it was needed, the path with the fewer hops would be preferable. Again opportunistic key management could suppress such a problem by the exchange of session keys for future usage.

4.4. Scenario 4

In this scenario, we assume rover A (security-source) wants to send 10 MB of sensitive data to the ground station B (security-destination). Node A (security source) has access to node's B (security destination) public key but they do not have a symmetric session key established. We compare three different cases. In the first case we encrypt the data with RSA (asymmetric encryption) and there is no need to establish a session key. In the second and third case we first establish a session key and then the data are encrypted with AES (symmetric encryption). Both algorithms (RSA and AES) are supported by the BSP protocol. In addition, in the second case we use the adoption of HMP protocol [15] (one-pass) for key establishment whereas in the third case we use the STS protocol (three-pass). The encrypted 10MB are incorporated in the last message of each protocol. From the three cases, one-pass symmetric encryption has the best performance. The simulation results are summarized in Table 1.

5. Discussion and Opportunistic Key Management

In DTNs a direct path between the source and destination does not always exist. Such a constraint makes routing in delay tolerant networks opportunistic. The main idea is that a custodian node will forward the data when there is an available path. The same idea can be adopted for key management, as a way to minimise the impact of such limitations of communication channel. Nodes that do not have access to security credentials could run key exchange protocols when the path is available and before the credentials are actually needed.

Therefore the node will store locally some session keys for future use. However, the best number of stored session keys depends on the available bandwidth, on the connectivity, on node storage availability and other policy specific security requirements that may be applicable.

Another usage of opportunistic key management could be opportunistic key confirmation. A plethora of key exchange protocols such as MQV, KEA etc. have 2-passes for key exchange and one more for key confirmation. Consider for instance a security-source node that does not have any stored session key and wishes to send encrypted data. In such case a key exchange protocol must run first. In order to minimize the imposed delay by the three passes, the node can initiate the first two passes for a key exchange and later when there is an opportunity it can send the key confirmation.

6. Concluding remarks, open research challenges and future work

In this paper we proposed a method for practically evaluating key exchange protocols in networks of adverse cation conditions. Our tool is designed to accommodate a plethora of interactive protocols for key exchange and/or key establishment. The proposed tool was realized in the form of a protocol parser that can be used either as a decision making tool running on a DTN node or a method for protocol evaluation in DTN environments. We confirmed for example that a node lacking availability of a certain cryptographic credential could be a prohibitive state in such networks. On this end, we introduced the concept of opportunistic key management, where such problem is minimized.

Future development of the parser and the simulation tools will involve providing the DTN layer with greater visibility into the details of the simulated security protocols thus enabling the implementation of more complex scenarios. Specifically, the protocol language will be enriched with a “cryptographic delay per byte of information” parameter, which will be passed on to the DTN agent. Data corresponding to actual user payload, as opposed to key exchange protocol messages, will be encrypted or decrypted at the DTN layer, inducing delay proportional to their size. In a simple scenario, application data will be encrypted at the originating node, or upon entering an untrusted network segment, and decrypted at the final destination. In the general case, a bundle may be encrypted/decrypted an arbitrary number of times when traversing different security domains throughout the network.

The enhancements mentioned above will enable evaluation of scenarios that build on scenario 4 of Section 4. Cryptographic delay of the payload will be combined with the delay introduced by the key exchange phase, highlighting the trade-off between the key exchange and the data transmission delay for different protocols and payload sizes. An example of such a trade-off is using computationally expensive public key cryptography while avoiding the key setup phase, versus employing more efficient symmetric algorithms that require key setup. Another example may involve the use of known keys for bundle decryption and re-encryption at intermediate, security domain gateway nodes, versus negotiating new end-to-end keys.

Adverse networking conditions, such as long delays and intermittent connectivity, make routing in DTNs a complex problem that heavily depends on the idiosyncrasy of each network setting. Security considerations render this problem even more challenging. Secure communication does not exclusively depend on contact opportunities, as in Contact Graph Routing [24], but also on key state information and key exchange protocol characteristics. Our work aims at gaining insight into the security aspects of DTN routing, in order to allow for educated, security-aware routing decisions, based on security-oriented network metrics.

References

[1] L. Wood, W. Eddy, P. Holiday, A bundle of problems, in: Aerospace conference, 2009, pp. 1–14.

[2] S. Farrell, S. Symington, H. Weiss, P. Lovell, Delay-tolerant networking

security overview draft-irtf-dtnrg-sec-overview-06, internet-draft (2009).
URL http://tools.ietf.org/html/draft-irtf-dtnrg-sec-overview-06

[3] S. Symington, S. Farrell, H. Weiss, P. Lovell, Bundle security protocol specification, Rfc6257 (2011).
URL http://tools.ietf.org/html/rfc6257

[4] S. Birrane, Streamlined bundle security protocol specification draft-irtf-dtnrg-sbsp-00, internet-draft (2013).
URL http://tools.ietf.org/html/draft-irtf-dtnrg-sbsp-00

[5] S. Burleigh, Dtn security key management - requirements and design draft-templin-dtnskmreq-00, internet-draft (2015).
URL https://tools.ietf.org/html/draft-templin-dtnskmreq-00

[6] A. Seth, S. Keshav, Practical security for disconnected nodes, in: Proceedings of the First International Conference on Secure Network Protocols, NPSEC'05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 31–36.
URL http://dl.acm.org/citation.cfm?id=1897159.1897165

[7] N. Asokan, K. Kostiainen, P. Ginzboorg, J. Ott, C. Luo, Applicability of identity-based cryptography for disruption-tolerant networking, in: Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking, MobiOpp '07, ACM, New York, NY, USA, 2007, pp. 52–56. doi:10.1145/1247694.1247705.
URL http://doi.acm.org/10.1145/1247694.1247705

[8] A. Kate, G. M. Zaverucha, U. Hengartner, Anonymity and security in delay tolerant networks, in: SecureComm, 2007, pp. 504–513.

[9] X. Lv, Y. Mu, H. Li, Non-interactive key establishment for bundle security protocol of space dtns, Information Forensics and Security, IEEE Transactions on 9 (1) (2014) 5–13. doi:10.1109/TIFS.2013.2289993.

[10] P. Edelman, M. Donahoo, D. Sturgill, Secure group communications for delay-tolerant networks, in: Internet Technology and Secured Transactions (ICITST), 2010 International Conference for, 2010, pp. 1–8.

[11] G. Xu, X. Chen, X. Du, Chinese remainder theorem based dtn group key management, in: Communication Technology (ICCT), 2012 IEEE 14th International Conference on, 2012, pp. 779–783. `doi:10.1109/ICCT.2012.6511309`.

[12] J. Zhou, M. Song, J. Song, X.-w. Zhou, L. Sun, Autonomic group key management in deep space dtn, Wireless Personal Communications (2013) 1–19`doi:10.1007/s11277-013-1505-1`.

[13] W. L. Van Besien, Dynamic, non-interactive key management for the bundle protocol, in: Proceedings of the 5th ACM Workshop on Challenged Networks, CHANTS '10, ACM, New York, NY, USA, 2010, pp. 75–78. `doi:10.1145/1859934.1859951`.
URL `http://doi.acm.org/10.1145/1859934.1859951`

[14] Z. Jia, X. Lin, S.-H. Tan, L. Li, Y. Yang, Public key distribution scheme for delay tolerant networks based on two-channel cryptography, J. Netw. Comput. Appl. 35 (3) (2012) 905–913. `doi:10.1016/j.jnca.2011.03.009`.
URL `http://dx.doi.org/10.1016/j.jnca.2011.03.009`

[15] S. Menesidou, V. Katos, Authenticated key exchange (ake) in delay tolerant networks, in: D. Gritzalis, S. Furnell, M. Theoharidou (Eds.), Information Security and Privacy Research, Vol. 376 of IFIP Advances in Information and Communication Technology, Springer Berlin Heidelberg, 2012, pp. 49–60. `doi:10.1007/978-3-642-30436-1_5`.

[16] A. Shikfa, M. Önen, R. Molva, Local key management in opportunistic networks, Int. J. Commun. Netw. Distrib. Syst. 9 (1/2) (2012) 97–116. `doi:10.1504/IJCNDS.2012.047898`.
URL `http://dx.doi.org/10.1504/IJCNDS.2012.047898`

[17] Y. Ding, X.-w. Zhou, Z.-m. Cheng, W.-l. Zeng, Efficient authentication and key agreement protocol with anonymity for delay tolerant networks, Wireless Personal Communications 70 (4) (2013) 1473–1485. `doi:10.1007/s11277-012-0760-x`.

[18] S. A. Camtepe, B. Yener, Key distribution mechanisms for wireless sensor networks: a survey, Tech. rep. (2005).

[19] R. Na, Y. Ren, Y. Hori, K. Sakurai, A generic evaluation method for key management schemes in wireless sensor network, in: Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, ICUIMC '11, ACM, New York, NY, USA, 2011, pp. 55:1–55:9. `doi:10.1145/1968613.1968680`.
URL `http://doi.acm.org/10.1145/1968613.1968680`

[20] B. Blanchet, Security protocol verification: Symbolic and computational models, in: P. Degano, J. Guttman (Eds.), Principles of Security and Trust, Vol. 7215 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 3–29.

[21] Pyparsing wiki home [online, cited 2015-03-18].

[22] Dtn agent for ns-2 [online, cited 2015-03-18].

[23] N. Bezirgiannidis, V. Tsaoussidis, Packet size and dtn transport service: Evaluation on a dtn testbed, in: Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on, 2010, pp. 1198–1205.

[24] S. Burleigh, Contact graph routing draft-burleigh-dtnrg-cgr-01, internet-draft (2010).
URL `http://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-00`

Table 1: Protocol Language

Function	Explanation	Usage Example
exp(x,y,p)	Exponentiation	exp(x,y,10)
	xˆy mod p	
hash(name,data)	Hash function	hash(SHA1,10)
	name is {SHA1,SHA256,SHA512,MD5}	
es(data,name,key)	Symmetric Encryption	es(10,AES,K)
	name is {AES,DES,TripleDES}	
	key is the symmetric key	
ea(data,name,key)	Asymmetric Encryption	ea(10,RSA,PubB)
	name is {RSA,ElGamal}	
	key is the symmetric key	
xor(data,data)	Exclusive OR	xor(10,15)
symkey(key,data)	Symmetric Key	symkey(K,10)
	key is the symmetric key	
privkey(key,data)	Private Key	privkey(PrivA,10)
	key is the private key	
pubkey(key,data)	Public Key	pubkey(PrivA,10)
	key is the public key	
cert(nn,data)	Certificate	cert(A,4)
	nn is node name	
keymaterial(material,data,key)	Key material	keymaterial(gˆra,10,K)
	material is used to create session key	
	key is the session key	
sig(data,key)	Signature	sig(10,PrivA)
	key is the private key	
id(nn,data)	Node identity	id(A,4)
	nn is node name	
mac(hash,key)	MAC	mac(hash(SHA1,10),K)
	hash is the returned value of hash function	
	key is the session key	
tstamp(data)	Timestamp	tstamp(4)
nonce(data)	Nonce	nonce(4)
plaintext(data)	Plaintext	plaintext(10)
mod(x,p)	Modulo	mod(x,10)
	x mod p	

data: is the input length in bytes

Table 2: Node Language

Function	Explanation	Usage Example
symkey=key	key is the symmetric key	symkey=K
privkey=key	key is the private key	privkey=PrivA
pubkey=key	key is the public key	pubkey=PubA
cert=key	key is the certificate	cert=A
connkey=privkey:pubkey	connection between private and public key	conn key=PrivA:PubA

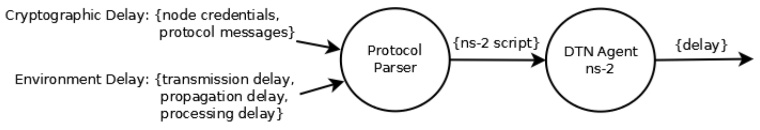

Cryptographic Delay: {node credentials, protocol messages}
Environment Delay: {transmission delay, propagation delay, processing delay}
Protocol Parser
{ns-2 script}
DTN Agent ns-2
{delay}

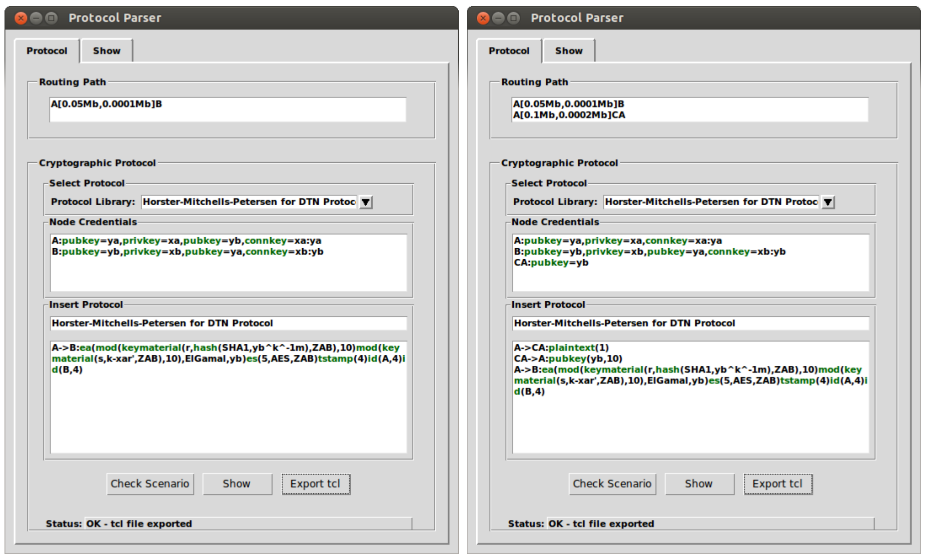
Figure 1: Procedure methodology

Table 3: Scenario 3

	Protocol Parser Input
Case 1	
Protocol Language	A→B:es(1024,AES,K)
Node Language	A:
	E:symkey=K
Path Language	A[0.05Mb,0.0001Mb]D[0.1Mb,0.0002Mb]E
Case 1 - after injection	
Protocol Language	A→KDC:plaintext(1)
	KDC→A:symkey(K,16)
	A→B:es(1024,AES,K)
Node Language	A:
	E:symkey=K
	KDC:symkey=K
Path Language	A[0.05Mb,0.0001Mb]D[0.1Mb,0.0002Mb]E
	A[0.1Mb,0.0002Mb]KDC
Case 2	
Protocol Language	A→B:plaintext(1024)
Node Language	A:
	E:
Path Language	A[0.05Mb,0.0001Mb]B[0.1Mb,0.0002Mb]C[0.1Mb,0.0002Mb]E

Table 4: End-to-end delay results

	case 1	case 2	case 3
Scenario 1	HMP	HMP without public key	-
	1801.7264 sec	5528.256 sec	-
Scenario 2	HMP	KEA/UM/MQV	STS
	1801.7264 sec	3726.52959 sec	5745.0528 sec
Scenario 3	Symmetric Encryption	Plaintext	-
	7331.7088 sec	5406.9054 sec	-
Scenario 4	RSA	HMP/AES	STS/AES
	2422328.8256 sec	2214639.2096 sec	2218567.8096 sec



```
Routing Path: A[0.05Mb,0.0001Mb]B
Protocol Library: Horster-Mitchells-Petersen for DTN Protoc
Node Credentials:
A:pubkey=ya,privkey=xa,pubkey=yb,connkey=xa:ya
B:pubkey=yb,privkey=xb,pubkey=ya,connkey=xb:yb
Insert Protocol: Horster-Mitchells-Petersen for DTN Protocol
A->B:ea(mod(keymaterial(r,hash(SHA1,yb^k^-1m),ZAB),10)mod(key
material(s,k-xar',ZAB),10),ElGamal,yb)es(5,AES,ZAB)tstamp(4)id(A,4)i
d(B,4)
Status: OK - tcl file exported
```

```
Routing Path:
A[0.05Mb,0.0001Mb]B
A[0.1Mb,0.0002Mb]CA
Protocol Library: Horster-Mitchells-Petersen for DTN Protoc
Node Credentials:
A:pubkey=ya,privkey=xa,connkey=xa:ya
B:pubkey=yb,privkey=xb,pubkey=ya,connkey=xb:yb
CA:pubkey=yb
Insert Protocol: Horster-Mitchells-Petersen for DTN Protocol
A->CA:plaintext(1)
CA->A:pubkey(yb,10)
A->B:ea(mod(keymaterial(r,hash(SHA1,yb^k^-1m),ZAB),10)mod(key
material(s,k-xar',ZAB),10),ElGamal,yb)es(5,AES,ZAB)tstamp(4)id(A,4)i
d(B,4)
Status: OK - tcl file exported
```

Figure 2: Scenario 1

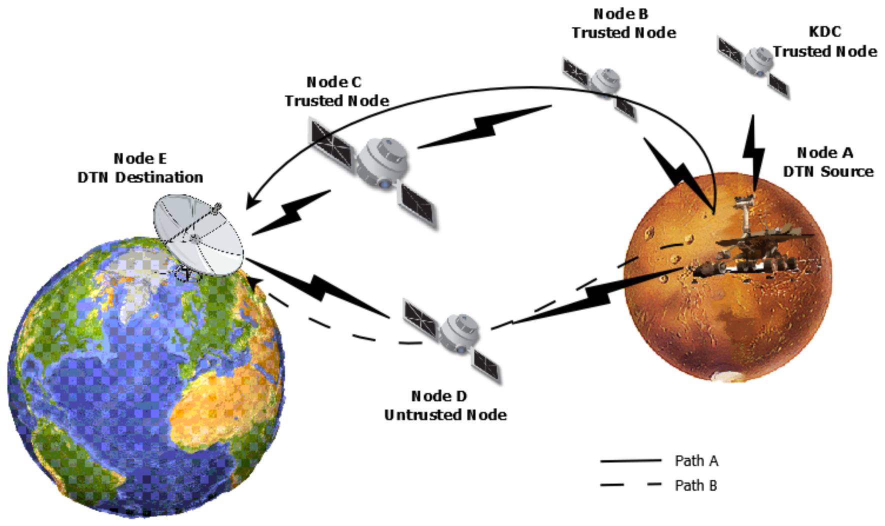


Figure 3: Scenario 3