



Research Center

NRC-TR-2007-007

Towards Securing Disruption-Tolerant Networking

N. Asokan^{1,2}, Kari Kostianen¹, Philip Ginzboorg¹, Jörg Ott², Cheng Luo²

¹Nokia Research Center, Helsinki

<http://research.nokia.com>

²Helsinki University of Technology, Finland

<http://www.tkk.fi>

March 21st, 2007

Abstract:

Most of the existing Internet applications are built on the assumption of immediate end-to-end connectivity. However, there are many scenarios, like messaging in sparsely populated areas, in which immediate connectivity is not available. Traditional approaches for communication security may not apply in such disruption-prone environments. Recently, the use of identity-based cryptography has been proposed as one way to help solve some of the security issues in disruption-tolerant networks (DTNs).

In this paper we discuss the security issues of DTNs, survey proposed solutions, and identify some improvements. In particular, we consider the applicability of identity-based cryptography as a solution for delay tolerant networking security, investigate how security in DTNs can be bootstrapped from existing large-scale security infrastructures, like the cellular communication security infrastructure, and present an improved scheme for authentication of fragments

Index Terms:

disruption-tolerant networking

identity-based cryptography

authentication

confidentiality

initializing security

Towards Securing Disruption-Tolerant Networking

N. Asokan, Kari Kostinen, Philip Ginzboorg
Nokia Research Center, Helsinki
{n.asokan, kari.ti.kostiainen, Philip Ginzboorg}@nokia.com

Jörg Ott, Cheng Luo
Helsinki University of Technology
{jo, cluo}@netlab.tkk.fi

Abstract

Most of the existing Internet applications are built on the assumption of immediate end-to-end connectivity. However, there are many scenarios, like messaging in sparsely populated areas, in which immediate connectivity is not available. Traditional approaches for communication security may not apply in such disruption-prone environments. Recently, the use of identity-based cryptography has been proposed as one way to help solve some of the security issues in disruption-tolerant networks (DTNs).

In this paper we discuss the security issues of DTNs, survey proposed solutions, and identify some improvements. In particular, we consider the applicability of identity-based cryptography as a solution for delay tolerant networking security, investigate how security in DTNs can be bootstrapped from existing large-scale security infrastructures, like the cellular communication security infrastructure, and present an improved scheme for authentication of fragments

1 Introduction

Many of the popular applications on the Internet today are built on the assumption of *immediate end-to-end* reachability: the ability to send a message to a peer, or a supporting server, and get back a response practically instantaneously. The only remaining notable exception to this assumption is e-mail, where the simple mail transfer protocol (SMTP) does allow messages to be stored and forwarded. But even though e-mail transport does not make the immediate end-to-end reachability assumption, services around e-mail do: for example, before sending a confidential e-mail, the mail user agent may expect to download the recipient's public key from a server; an e-mail may contain links to embedded images which the mail user agent expects to download when the user opens the e-mail message.

While this assumption holds true for most Internet communication today, there are many use cases for which it is invalid: space communication and networking in sparsely populated areas are examples of such *delay- and disruption-tolerant networking* (DTN) [5].

In this paper we discuss the security issues related to DTNs, survey proposed solutions, and identify some improvements. In particular we consider the applicability of identity-based cryptography as a solution for delay tolerant networking security, investigate how security in DTNs can be bootstrapped from existing large-scale security infrastructures, like the cellular communication security infrastructure, and present an improved scheme for authentication of fragments.

We start this paper by describing a motivating example use case (Section 2), identifying security issues for DTN communication (Section 3), and reviewing related work (Section 4). Then we analyze the applicability of identity-based cryptography for DTN security (Section 5), discuss security initialization (Section 6), and present a new method for authentication of fragments (Section 7). Finally, we discuss our current work (Section 8) and conclude (Section 9).

2 Use case

Alice is an enthusiastic backpacker travelling for a month in different countries. Using her smartphone she takes photographs, records video clips, and checks her emails. In her journey she will visit various locations, both densely populated cities where she can connect to the Internet using her phone and distant villages where direct Internet connectivity is not available.

Alice wants to keep in touch with her friends at home as much as possible, and send them pictures, video clips and messages even from the distant villages. This is made possible with the help of like-minded travellers that Alice meets during the trip. Alice has just shot several cool videos which she wants to send to her friends, but there is no Internet connectivity available. Another backpacker Bob is soon going to board a bus to the nearest small town later that day. Alice's smartphone transfers the video messages to Bob's device using a short range wireless connection and the messages travel with Bob to the nearest town.

Unfortunately, there is no Internet connectivity in the town where Bob arrives to and his device cannot forward Alice's messages to the Internet. Instead, he meets another backpacker Carole, who is taking the train to a bigger city. Bob's device transfers Alice's messages to Carole's device. When Carole reaches the big city with Internet connectivity, her device forwards Alice's messages to Alice's friends through Internet.

3 Security issues in DTN environments

The example scenario described in the previous section is based on the assumption that Bob and Carole are willing to store and forward messages received from Alice. Since mobile devices have limited resources in terms of battery life, memory, storage capacity and network bandwidth, Bob and Carole would probably like to be in control of their level of participation to this system. For example, Bob might only want to forward messages received from the people in his phonebook while Carole might have an even more strict personal forwarding policy.

This kind of policy-based routing requires that the devices are able to *authenticate* both the original sender and the intermediate sender of the message, and verify the *integrity* of received messages. Otherwise, an adversary might cause unwanted consumption of limited resources.

Some of the communication in DTN might also be personal in nature. For example, Alice might not want Bob or Carole to see the pictures she is sending to her friends at home. Thus, we need *confidentiality* of end-to-end communication.

Due to the special nature of DTN environments the traditional security mechanisms are not always applicable [15]. For example, end-to-end confidentiality using traditional encryption mechanisms requires the sender to know a recipient-specific encryption key. If Alice does not already have encryption keys of her friends, and has no immediate connectivity to the recipients or a supporting server, she will not be able to send a private message to the recipient.

A second issue is *fragmentation*. Messages in delay-tolerant networks may be big. For example, if Alice would like to send a lengthy video clip to her friend, the message size could be tens of megabytes. When a network link, such as temporary wireless connection to another traveller, becomes available for a short period of time, it may not be possible to send the entire message at once. The original sender or an intermediary device might have to fragment the message into smaller pieces, and send only partial message through the current link and rest of the message through another link later to make the best use of limited resources.

The intermediary and final recipients should be able to verify the origin and integrity of all received fragments. Traditional authentication mechanisms, such as calculating a hash over the entire message body, signing the hash and attaching the signature to the message are not optimal when fragmentation of messages is needed.

Another issue that should be taken into account is *privacy*. Alice might know Bob personally and trust him, but she might not want other intermediary devices to know anything about her communication. The traditional privacy techniques, such as the use of pseudonyms or successive layered encryption (sometimes referred as Onion Routing [9]), do not interact well with policy-based routing.

4 Related Work

In this section we review existing work related to security issues in delay-tolerant networks.

4.1 DTN security overview

Farrell and Cahill review the current state of DTN security work in [6]. They identify that the main threats for DTNs are 1) modification of messages (or “bundles”) in transit for malicious purposes e.g. for masquerading attacks, 2) unauthorized use of scarce DTN resources, and 3) denial of service e.g. by mounting amplification attacks [13] in which a malicious node uses the replay properties of DNT protocols and unaware nodes for generating large amounts of traffic to the network.

Farrell and Cahill propose that a security architecture is needed in which security services can be provided both on hop-by-hop and end-to-end basis, and additionally between two intermediary nodes in the middle of a route. For example a gateway of a sensor network could function as a *security-sender* by encrypting the bundles sent by the computationally less powerful sensor nodes to Internet residing recipients. In a similar fashion, an Internet connected DTN node could function as a *security-destination* and authenticate all received bundles using its ability to fetch public keys and check certificate revocation lists from the Internet before forwarding the bundles to the real recipient which might not have Internet connectivity.

Another identified requirement is that all DTN nodes should provide atleast very rudimentary support for policy-based routing. For simple DTN nodes the policy could be just to forward all received bundles while the computationally more powerful nodes could perform more complicated policy decisions based on the current resource consumption state of the network, the authentication of the bundle, the time-to-live information attached to the bundle, the route the bundle has travelled so far etc.

Farrell and Cahill note that several open issue remain in DTN security. The level of flexibility that should be provided is one of the open issues; while the security architecture should provide security services in various ways (hop-by-hop, end-to-end, security-sender to security-recipient) the implementation cost and level of complexity should not rise too high, since typically complicated solutions are not secure in practice. Another currently open issue is key management. The authors argue that most of the existing solutions are not suitable for DTNs (e.g. TLS [3] handshake requires too many roundtrips). Using fixed keys is feasible for experimental work, but it does not scale to real world DTN deployments. The authors suggest that maybe key management solutions developed for ad-hoc networks, such as the Resurrecting Duckling model [16], could be utilized in DTN environments.

4.2 Bundle security protocol specification

The Delay Tolerant Networking Research Group (DTNRG) ¹ has produced an Internet draft describing a bundle security protocol specification [17] and an additional draft [7] explaining the rationale for the design choices made in the specification. The specification describes three IPsec [10] style security headers that can be added to bundles to provide different security services. Bundle Authentication Header (BAH) provides authentication for single hop by adding a message authentication code or a signature to the bundle. The Payload Security Header (PSH) is used to provide end-to-end authentication in a similar fashion. Confidentiality Header (CH) is used to encapsulate encrypted payload.

Each security header contains the security-source and the security-destination information and a ciphersuite. The ciphersuite defines the algorithms that should be used to process the received security headers. The security-sender and the ciphersuite information together determine the keys that should be used. Different combinations of these three security headers can be used simultaneously.

4.3 Authentication of fragments

The simple way to integrity protect and authenticate bundles is to calculate a hash of a bundle, sign this hash using the senders private key and attach both the hash and the signature to the end of the bundle. Unfortunately, this approach does not work well in DTNs where fragmentation of bundles is often needed. Let us assume that a device is sending a large bundle in fragments when the underlying connection breaks. The receiver has not yet received the entire message and thus it cannot verify the hash

¹<http://www.dtn.org>

and authenticate any of the received fragments. The sender cannot just send the remaining fragments via another path in the network, instead it has to either wait for the recovery of the broken link or send the whole bundle again using another path. Neither of the alternatives are optimal.

One solution to this problem is the so called *toilet paper* approach that has come up at the DTNRG mailing list discussions [4]. The main idea is to make each fragment self-authenticating by attaching a signed hash to the end of each fragment separately. This approach has two limitations: 1) since signatures are typically large, a considerable amount of extra traffic is generated, and 2) since both generating and verifying a signature are computationally extensive operations, a serious performance overhead is introduced.

Partridge presents a few solution proposals for this fragment authentication problem in [12]. The first proposal is to use *cumulative authentication* in which each fragment f_i is authenticated by calculating a hash over all the previous fragments including the current fragment $f_1 \dots f_i$. The amount of work required from the receiver is less than in toilet paper approach, since only one signature per a *set* of received fragments has to be verified. On the other hand, this approach does not reduce computation work of sender or the amount of traffic, since a signature has to be generated and added to each fragment. In addition, this approach assumes that fragments are received in order which might not be the case always.

The second proposal in Partridge's paper is to authenticate fragments using *function definitions*. In this approach the purpose is to find a suitable iv for an authentication function A so that for each fragment $f_1 \dots f_n$ the output value of the authentication function calculated over the iv and the fragment f_i is that fragment's sequence number i :

$$A(iv, f_1) = 1, A(iv, f_2) = 2, \dots, A(iv, f_n) = n$$

If it is possible to find an authentication function A which has a small representation and a small iv it will become more efficient to sign the representation of the function and iv and send them instead of sending separate signed hashes for each fragment. Partridge shows that a normal hash function, such as MD5 [14], can be used as an authentication function if the iv is large (actually, as large as the concatenation of n hash values), and thus using this function is not more efficient than the toilet paper approach. Finding a more efficient authentication function is left as an open issue.

4.4 Using identity-based cryptography for DTN security

Identity-based cryptography (IBC) [2] is a relatively new cryptographic method that enables message encryption and signature verification using the public identifier, such as email address, of the target as a key. An IBC system consists of principals (e.g. message senders and recipients) and a commonly trusted third party called the private key generator (PKG). At system initialization phase the PKG generates system-wide public parameters PP and a corresponding master secret key S_{PKG} . Using S_{PKG} and an identifier id_P the PKG can generate a private key S_P for principal P. At this point the PKG must verify that the principal really is allowed to use this particular identifier id_P , and a confidential communication channel is needed to securely deliver the private key S_P to the principal. A message can be encrypted to P using PP and id_P . P can decrypt the resulting ciphertext using the S_P it received from PKG. In similar fashion, P can sign messages using S_P and other principals can verify the signature using id_P and PP .

Seth et al. [15] have proposed a security architecture for DTNs based on IBC. They argue that traditional PKI is not well suited for disconnected environments, such as DTNs, since access to online servers for fetching public keys and checking certificate revocation lists cannot be assumed. They use a variation of IBC known as hierarchical identity based cryptography (HIBC) [8] in which different regions have sub-regions each maintaining their own PKGs. The messages sent from user of one PKG to a user of another PKG are authenticated and protected using the trust relations between PKGs and standard techniques of HIBC. The identifier of a principal can be based on existing well-known identifiers such as e-mail addresses. Seth et al. do not describe how a PKG can verify whether a new principal does indeed have the right to a well-known identifier: for example, they assume that authorized distribution agents like kiosks can somehow authenticate principals and issue credentials in USB sticks which they can use to enroll with the PKG.

5 Applicability of identity-based cryptography

In this section we analyze the applicability of identity-based cryptography (IBC) for DTN communication.

5.1 Authentication and integrity

Messages in DTNs may need to be authenticated for several different reasons. Due to the limited resources intermediate nodes may want to use authentication as the basis for policy-based routing and forwarding (e.g. an intermediary node might only want to store and forward message from a pre-defined set of known senders). The recipient might also want to authenticate the originator for deciding how to interpret the contents.

The current DTN bundle security specification [17] supports authentication using a message authentication code, or a digital signature. These are sufficient in situations where the sender has pre-existing security associations with the various verifiers (intermediary nodes and final recipients). The specification could be easily extended so that messages also carry the necessary certificates using which a verifier can validate a digital signature.

Seth et al. [15] argue that certification revocation lists are unsuitable for DTNs because updates to these lists can be delayed excessively in a disconnected environment. Instead, they propose the use of IBC. In IBC, revocation is avoided by periodically refreshing the underlying identifiers, and hence the signing keys. Each signing key is valid for a short period (e.g., a day). An underlying identifier is constructed by concatenating the long-lived identifier with a description of the validity period: e.g., `alice@example.com:15-03-2007` to refer to the underlying identifier that should be used to encrypt messages for Alice on March 15, 2007. A verifier can check if the message is signed with a sufficiently recent signing key. Thus, instead of requiring the verifier to receive revocation lists in a timely fashion, IBC-based authentication schemes require the signer to receive fresh signing keys periodically.

A similar approach can also be used with traditional public key cryptography: the certificates issued to signing keys can be short-lived (e.g., valid for a day). The signer must periodically receive new certificates from the certification authority (CA), but the signing key itself may be long-lived. A verifier can check if the message is correctly signed and is accompanied by a sufficiently recent certificate. Thus we conclude that authentication needs in DTNs can be met without resorting to IBC, but through the judicious use of traditional cryptographic techniques instead. Note that when digital signatures are used for authentication, the sender can compute all the necessary authenticators even when there is no network connectivity.

Compared to IBC authentication, traditional signature based authentication increases message size slightly. Adding a certificate (or a chain of certificates verifying all the intermediary CAs from the root CA to the sender) increases the message size up to few kilobytes. However, if messages are relatively large, say at least hundreds of kilobytes, the overhead introduced by the certificate(s) is not significant.

5.2 End-to-end confidentiality

Confidentiality is achieved through the use of encryption. Unlike in the case of authentication, the sender may not be able to use traditional cryptographic techniques for encryption if there is no network connectivity at the time of sending the message. This is because with traditional cryptographic techniques, the sender needs to have a valid recipient-specific encryption key in order to encrypt the message for a certain recipient.

With this in mind, Seth et al. [15] use IBC for the encryption of DTN messages. A sender can compute an IBC encryption of a message for a recipient by just knowing the recipient's identity and common public system parameters. This allows the sender to construct the encryption even when there is no network connectivity.

In [7] Farrell et al. argue that the usage of IBC does not solve the difficulty of doing validity checking in DTNs because checking the validity of the IBC public parameters (*PP*) is equivalent to verifying a CA certificate in traditional public key cryptography. This is not a fair comparison. Public system parameters in IBC systems are system-wide parameters, comparable to *root* public keys in traditional public key systems: both are long-lived, and are the roots of trust. As such, their validity is not verified frequently. In traditional public key systems, the encrypting party needs to validate recipient-specific information (recipient's public key). With IBC, this burden is removed, and thus, with respect to the

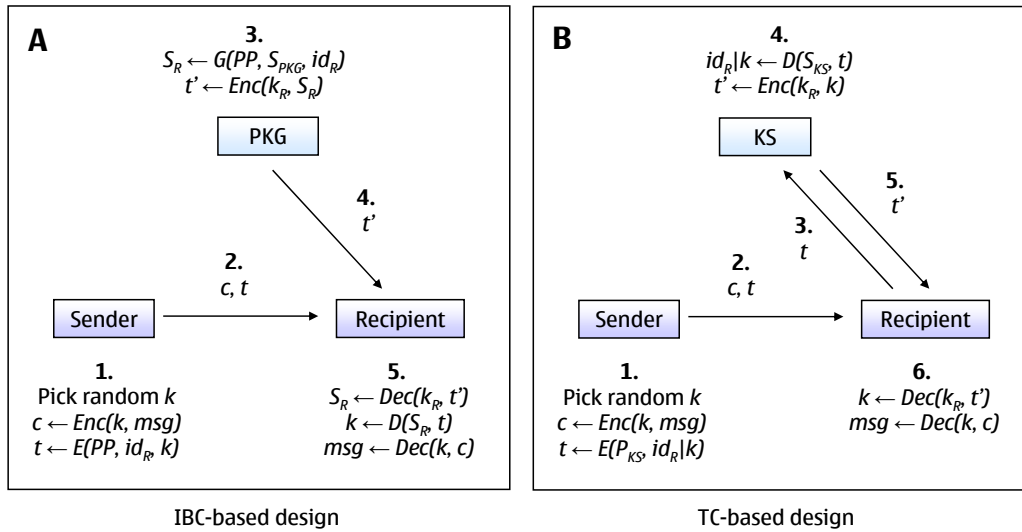


Figure 1: Private messaging in a disconnected environment using identity-based and traditional cryptography

validity checking issue, encryption using IBC does offer an advantage for DTNs compared to encryption using traditional public key cryptography in the usual manner.

But is the use of IBC really justified? Is it possible to get similar behaviour by using traditional cryptographic techniques in a way that is more suitable to disconnected environments?

We answer this question by constructing two different designs for *private messaging in disconnected environments*. One design is based on identity-based cryptography, while the other is based on traditional cryptographic techniques only. In both designs, a sender S should be able to send a message msg to a recipient R such that no unauthorized party can read the contents of this message. S should be able to compose the message and encrypt it without requiring online access to R or any server. In both systems, there is a fully trusted server which aids in secure messaging. We assume that each R has a long-lived security association with the server. In the simplest case, it can be a shared symmetric key k_R .

5.2.1 IBC-based design for private messaging

In this design the server is an IBC private key generator PKG. We assume that the PKG has IBC public parameters PP and the corresponding secret key S_{PKG} . PP and the identity of the recipient id_R are known to the sender ahead of time. The recipient and the PKG share a symmetric key k_R . We denote asymmetric IBC encryption and decryption with $E()$ and $D()$, IBC public parameters with PP , IBC private key generation with $G()$, and symmetric key encryption and decryption with $Enc()$ and $Dec()$.

The IBC-based design is illustrated in part A of Figure 1. First, sender S picks a random symmetric key k , encrypts the message msg using symmetric encryption and creates an envelope t that holds k using IBC encryption (step 1). Sender S sends both the ciphertext c and the envelope t to recipient R (step 2). To be able to decrypt the received ciphertext c , recipient R needs the right IBC private key issued by the PKG. Recipient R connects to PKG, which generates an IBC private key S_R for R . This private key is encapsulated into a recipient envelope t' (step 3). PKG sends the envelope t' to R (step 4). The recipient decrypts the envelope to get S_R , and uses it to recover the message encryption key k and then the message msg itself (step 5).

It should be noted that steps 3 and 4 can happen independently of steps 1 and 2. This means that a device can fetch a fresh IBC private key even *before* receiving any messages. All devices could be configured to fetch a fresh IBC private key the first time they are connected to network during a new key validity period. From then on, the devices could decrypt any received message encrypted during that key validity period without contacting PKG again. As we saw in Section 5.1, the validity period is encoded in the underlying identifier used for encryption. It may be a system-wide parameter, or a user- and/or application-specific parameter.

5.2.2 TC-based design for private messaging

Now, we consider how to implement the same usage scenario using only traditional cryptography (TC). We assume that instead of the PKG there is a key server KS with a public key P_{KS} and the corresponding secret key S_{KS} . The public key P_{KS} and the identity of the recipient id_R are known to the sender S ahead of time. S does not know any recipient-specific public key for R (in fact R may not even have a public key of its own). $E()$ and $D()$ denote classical public key encryption and decryption operations. The rest of the system is same as in the previous case.

The TC-based design is illustrated in part B of Figure 1. First, sender S picks a random symmetric key k , and computes a ciphertext c and an envelope t that holds both id_R and k using traditional public key encryption and public key of KS (step 1). S sends envelope t and ciphertext c to recipient R (step 2). R forwards the envelope to key server (step 3) which decrypts envelope t and constructs a recipient envelope t' that holds k (step 4). Key server sends t' to R (step 5) and recipient first recovers k from t' and then uses it to recover the message msg from the ciphertext (step 6).

It should be noted that in this design, steps 3, 4 and 5 *must* happen after steps 1 and 2. Thus the recipient needs to have network connectivity at the time of reception or otherwise the decryption is delayed until connection to key server can be finally established.

5.2.3 Comparison of IBC- and TC-based designs

As shown in the previous section, the TC-based design sets more demands on recipient's network connectivity: the recipient must have access to the key server at the time of reception or otherwise the message decryption is delayed. In the IBC-based design the recipient can fetch the private IBC key prior to reception, and thus network connectivity is not necessarily needed at the time of reception and decryption.

To analyze the computational load caused by these two designs assume that there are s senders in the system sending m messages each addressed to d different receivers on the average. In the same system there are r receivers each receiving e messages on the average. The total number of messages sent and received is the same $smd = re$. The computational loads are presented in Table 1.

Work done by	IBC-based	TC-based
Server	r IBC key generation r symmetric encryptions	sm private key decryptions smd symmetric encryptions ¹
Sender	md IBC encryptions m symmetric encryptions	m public key encryptions m symmetric encryptions
Receiver	e IBC decryptions $1 + e$ symmetric decryptions	$e + e$ symmetric decryptions ²

Table 1: Comparison of IBC- and TC-based designs for private messaging

¹If several envelopes are sent to the server in step 3, the server can package all decryption keys belonging to the same receiver into the same recipient envelope, resulting in fewer symmetric key encryptions. In the best case all decryption keys belonging to the same receiver could be packaged into a single envelope, requiring the server to perform only r symmetric encryptions in total.

²If the server packages multiple decryption keys into same envelope, the receiver has to do fewer decryptions. In the best case number of required decryptions is $1 + e$ (one decryption for the envelope containing all decryption keys and e message decryptions).

The IBC-based design requires less work from the server. The number of required IBC key generations is proportional to the number of receivers in the system r while in the TC-based design the server has to decrypt each received envelope and thus the number of required private key decryptions is proportional to the number of senders *and* the number of sent messages sm .

On the other hand, the IBC-based design requires more work from the sender. In the TC-based design one message addressed to multiple recipients requires only one public key encryption while in the IBC-based design the sender has to create an envelope for each separate recipient and thus md IBC encryptions are required. The IBC-based design requires also more work from the receiver. Each received message has to be IBC decrypted while in the TC-based design only computationally less intensive symmetric decryptions are needed.

As a conclusion, the IBC-based design should be preferred since it has less strict requirements on recipient's network connectivity and it imposes considerably less load on the server. Only for the most computationally restricted DTN clients, such as sensor nodes, the TC-based design might be preferable since the IBC-based design requires performing up to d IBC encryptions per every sent message. For typical DTN clients, such as laptops, PDAs and phones, this is not a problem.

6 Initializing security

In our discussion so far we have assumed that both senders and recipients have a previously established security association with a trusted server, such as PKG. Initializing secure connections is a difficult problem. Security initialization is *expensive* both in terms of money as well as time, and it is often economically attractive to bootstrap security from an existing infrastructure [1].

Seth et al. [15] allow for bootstrapping by permitting the use of well-known identifiers like e-mail addresses. When using such an external name space, the PKG must have the means to verify that a principal wanting to enroll using a well-known identifier does in fact have the right to claim that identifier. Seth et al. [15] do not spell out how such a check could be done. In particular, in the case of disconnected users, enrollment is done via authorized distribution agents kiosks. The identity of a new enrolling user is verified at the kiosk before the kiosk maintainer issues the credentials to the new user on a USB stick. This would be hard to implement securely in practice. Consider an example scenario in which Alice wants to enroll to the system using an email address `alice@example.com` as her identifier. How can the kiosk maintainer verify that Alice really is the owner of this particular email address when she is *disconnected*? For connected users, the PKG could use some form of return routability check to verify ownership: e.g., by sending a secret enrollment key by e-mail to the claimed e-mail address. This has two problems: first, it is difficult to secure the communication paths; and second, subsequent revocation of the well-known identifier will not be noticed by the PKG.

6.1 Bootstrapping from the cellular authentication infrastructure

An alternative is to bootstrap from the cellular security infrastructure. This infrastructure consists of credentials already provisioned by cellular operators to cellular subscribers e.g. in the form of SIM cards, and roaming agreements among different cellular network operators. This is by far the most widely deployed authentication system, with more than one billion users and hundreds of participating cellular operators. The latest 3G specifications include a feature called Generic Authentication Architecture (GAA) [11]. The main idea behind GAA is to use the cellular security infrastructure to initialize secure connections between mobile devices and *application servers*. Bootstrapping using GAA is illustrated in part A of Figure 2. First, the mobile device and a *bootstrapping server* of the cellular operator engage in the usual cellular network authentication protocol. As a result, the mobile device and the bootstrapping server share a master session key k_M (step 1). Later, when a mobile device needs a secure connection to an application server, the mobile device can derive a server-specific shared session key k from the master session key and the identifier of the application server (step 2). The mobile device may now use the application server using k (step 3) and the application server may obtain the same key from the bootstrapping server (step 4).

The DTN CA and PKG servers will act as GAA application servers, run either by the cellular operator or independent third parties that have service agreements with the operator. The CA will use GAA to authenticate the enrollment of public keys, and issue short-lived subscriber certificates. The PKG will use GAA to encrypt IBC private keys for devices. A principal is identified by a well-known identifier (e.g., email address or mobile phone number) which is securely bound to a cellular identifier. Revocation of this identifier will be automatically reflected in the DTN security infrastructure, since the device will no longer be able to receive short-lived certificates or IBC private keys.

Although most people have mobile phones, not all DTN clients, such as laptops and PDAs, have SIM cards needed for cellular authentication. This problem can be solved by using a short range wireless connection, such as Bluetooth. One example solution is illustrated in part B of Figure 2. We assume that the user has already established a security association between his non-cellular devices, such as laptops, and his mobile phone, e.g. by doing Bluetooth pairing. When a non-cellular device needs a secure connection to an application server, such as CA or PKG, it may send a request to its paired mobile

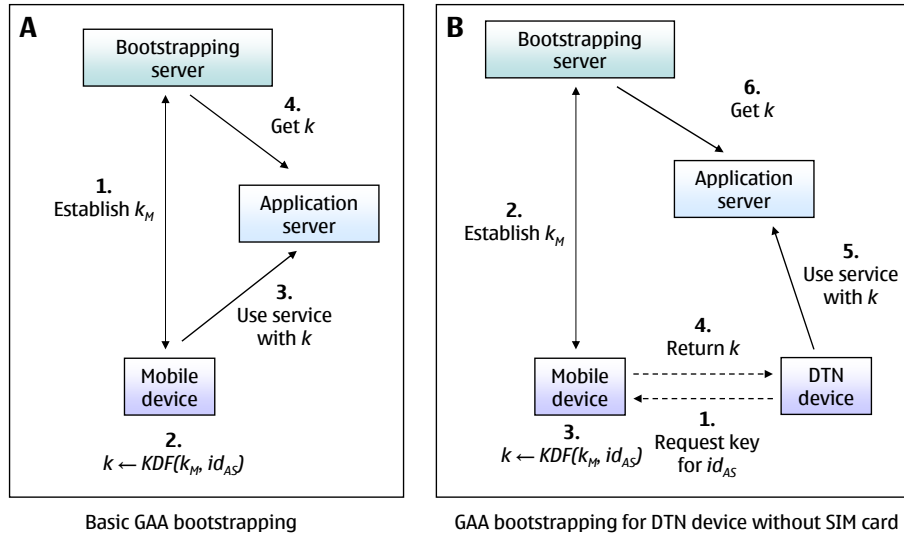


Figure 2: Initializing secure connections using Generic Authentication Infrastructure (GAA)

phone. (step 1). The mobile device may now do the normal cellular authentication (step 2) and derive the application server specific key (step 3). The mobile phone sends this key back to the non-cellular device over the short range wireless connection (step 4) and the non-cellular device may now use the service (step 5) and the server may obtain the same key (step 6).

6.2 Cross-domain operation

In the previous sections we have assumed that all devices have a trust relationship with the *same* trusted third party (CA or PKG). In practice, depending on a single commonly trusted entity is not a flexible solution. Seth et al. [15] have proposed the use of hierarchical identity based cryptography (HIBC) [8] for DTNs to overcome this limitation, but using hierarchical identity-based cryptography is not necessary when bootstrapping from the cellular authentication infrastructure, because cellular operators already have roaming agreements intended to enable cross-domain operation.

Consider the following scenario. Alice is subscriber of operator A which has a service agreement with PKG_A . Alice may bootstrap secure connections to this server and thus send and receive confidential messages to and from all other subscribers of operator A . Bob is a subscriber of another operator B and by default Bob uses PKG_B . When Alice wants to send an encrypted message to Bob she cannot use the public parameters of PKG_B if she does not have connectivity at the time of encrypting and she cannot fetch these parameters from the network. Instead Alice encrypts the message using public parameters of PKG_A . To decrypt the message Bob needs to fetch a private key from PKG_A . This is possible due to the roaming agreement between the two operators. Bob may bootstrap a secure connection to PKG_A using bootstrapping the server of operator B which has a roaming agreement with operator A which in turn has a service agreement with PKG_A . Thus hierarchical identity-based crypto is not really needed.

7 Authentication of fragments

In Section 3, we saw that intermediate nodes may need to fragment bundles, and that this is in conflict with the need to verify the sender of a fragment. One proposal to address this problem is the *toilet paper* approach (see Section 4.3). However, a straight-forward application of the toilet-paper approach of attaching a signature to each fragment is considered undesirable because of the message size and computational overheads [7]. Let us call this the “list of signatures” approach.

We present an alternative approach based on constructing a binary hash tree. The security-sender prepares n fragments (for simplicity, assume that n is a power of 2) f_i of a bundle as in the toilet-paper approach and calculates the first set of n hashes $h_i = h(f_i)$, $i = 1 \dots n$. He then calculates the next set of

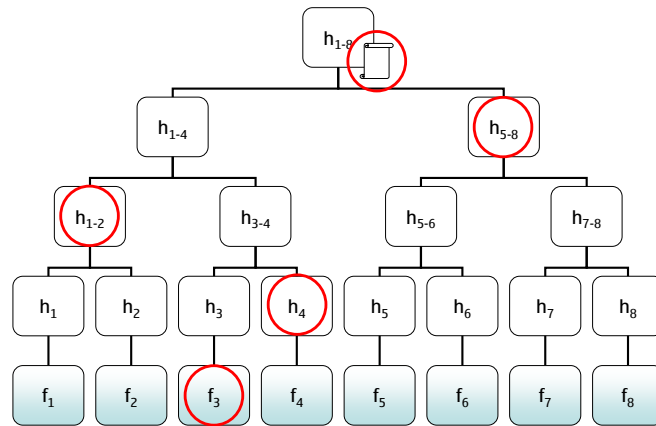


Figure 3: Binary hash tree

$n/2$ hashes by combining the first set of hashes in pairs: $h_{h_{i-i+1}} = h(h_i, h_{i+1})$, $i = 1, 3, \dots, n-1$. He then repeats the process until he calculates a single hash h_{1-n} which represents the entire set of fragments. Figure 3 illustrates the binary hash tree thus formed for a 8-fragment bundle. The final hash is signed by the security-sender. To authenticate each fragment a verifier needs the signature of the top hash, and a set of $\log(n)$ hashes, one from each level of the tree. Figure 3 shows an example: to be able to authenticate f_3 , the verifier needs the set of hashes h_4, h_{1-2} and h_{5-8} , and the signature on the top hash h_{1-8} , in addition to the fragment f_3 itself. A verifier can start from the contents of f_3 and calculate each hash in the path from the f_3 leaf node to the root node, and finally verify if the accompanying signature is a valid signature on the calculated root hash value.

Notice that the signature needs to be transferred only once between a forwarding node and a receiving node. Each fragment has to carry $\log(n) - 1$ hash values for verifying itself. The security-sender has to compute 1 signature and $2n - 1$ hash operations. In contrast, in the straight-forward toilet-paper approach of signing every fragment, each fragment has to carry one signature. The security-sender has to compute n hashes and n signatures. The differences are illustrated in Table 2

	List of signatures approach	Binary hash tree approach ¹
Work by security-sender	n hashes + n sigs.	$(2n - 1)$ hashes + 1 sig.
Work by verifier	m hashes + m verifs.	$\leq m(\log(n) - 1)$ hashes + 1 verif.

Table 2: Comparison of strategies for fragment authentication

¹Total number of fragments = n , Fragments transferred in a session = m

8 Further issues

Other methods of validating senders: In Section 5.1, we argued that one rationale for authenticated DTN communication is to enable policy-based forwarding: intermediate nodes decide whether to store and forward the message based on local state, such as remaining battery life, and the sender's identity. Forwarding policies may be based on other forms of authentication as well. For example, a sender who does not want to reveal his identity may opt to use trusted hardware on his device to provide a *remote attestation* certifying the integrity of the DTN client's software and hardware platform.

Proposed changes to bundle security protocol specification: Above, and in Sections 5.1 and 7 we discussed different ways by which intermediate nodes, as well as the final recipient, may validate the sender of a DTN message. Each of these requires the addition of multiple pieces of authenticators (e.g., a certificate and signature). The latest draft (version -02) of the Bundle Security Protocol (BSP) [17] does not support this. Currently, the security result field of BAH or PSH headers is a length-value encoded pair

that may contain *either* a message authentication code or a signature calculated from the payload. The ciphersuite-ID in the the header should completely specify how the “value” field is interpreted. However, the specification does allow multiple authentication headers to be linked together using a correlator.

We propose that the BAH and PSH headers should be able to contain *multiple* type-length-value encoded security result items.

9 Conclusion

Having analyzed the applicability of identity-based cryptography in addressing the security needs for DTN communication we conclude that (a) compared to using traditional cryptography IBC does not provide any significant improvement in authentication, and (b) IBC does lend to a more efficient solution for end-to-end confidentiality, in terms of server load and network connectivity requirements for recipients.

Bootstrapping from the cellular security infrastructure seems to hold promise in addressing the initialization, key management, and cross-realm operation aspects of DTN security.

In our current work, we are implementing the bundle security protocol specification with enhancements to support multiple authenticators. We are also designing and implementing Bluetooth-based protocol using which a non-cellular device can obtain a subscriber certificate from the GAA infrastructure with the help of a cellular device.

Several open issue remain:

- Is there a more space-efficient approach for authenticating fragments?
- How can we incorporate remote attestation as one means of validating the security-sender?

We hope to address these issues in future work.

References

- [1] N. Asokan and Lauri Tarkkala. Issues in initializing security. In *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, pages 460 – 465. IEEE Press, December 2005.
- [2] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001, Advances in Cryptology*, number 2139 in Lecture Notes in Computer Science, pages 213–229. Springer-Verlag, August 2001.
- [3] Tim Dierks and Christopher Allen. The TLS protocol version 1.0. RFC 2246, IETF, January 1999.
- [4] DTNRG. Delay tolerant networking research group: `dtn-interest` mailing list archive, April 2005. Available from <http://mailman.dtnrg.org/pipermail/dtn-interest/2005-April/>.
- [5] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of SIGCOMM 2003*, August 2003.
- [6] Stephen Farrell and Vinny Cahill. Security considerations in space and delay tolerant networks. In *Proc. 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, July 2006.
- [7] Stephen Farrell, Susan Symington, and Howard Weiss. *Delay-Tolerant networking security overview*. IRTF, DTN research group, October 2006. Draft version -02; expires in April 2007.
- [8] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In *8th International Conference on the Theory and Application of Cryptology and Information Security*, December 2002.
- [9] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2), February 1999.
- [10] Stephen Kent and Randall Atkinson. Security architecture for the Internet Protocol. RFC 2401, IETF, November 1998.

- [11] Pekka Laitinen et al. Extending cellular authentication as a service. In *Proceedings of The First IEE International Conference on Commercialising Technology and Innovation*, September 2005. <http://www-admin.iee.org/OnComms/PN/communications/062%20-%20P%20Ginzboorg.pdf>.
- [12] Craig Partridge. Authentication for fragments. In *HotNets-IV, The Fourth Workshop on Hot Topics in Networks*, 2005. Available from <http://www.acm.org/sigs/sigcomm/HotNets-IV/program.html>.
- [13] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *Computer Communication Review*, 31(3), July 2001.
- [14] Ronald Rivest. The MD5 message-digest algorithm. RFC 1321, IETF, April 1992.
- [15] Aaditshwar Seth, Urs Hengartner, and Srinivasan Keshav. Practical security for disconnected nodes. In *First Workshop on Secure Network Protocols (NPsec)*, November 2005. Revised 2006 version of the NPsec paper http://www.cs.uwaterloo.ca/~a3seth/practical_security_v2.pdf.
- [16] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *The 7th International Workshop on Security protocols*, April 1999.
- [17] Susan Symington, Stephen Farrell, and Howard Weiss. *Bundle Security Protocol Specification*. IRTF, DTN research group, October 2006. Draft version -02; expires in April 2007.