# Applicability of Identity-Based Cryptography for Disruption-Tolerant Networking

N. Asokan
Nokia Research Center
n.asokan@nokia.com

Kari Kostiainen
Nokia Research Center
kari.ti.kostiainen@nokia.com

Philip Ginzboorg
Nokia Research Center
philip.ginzboorg@nokia.com

Jörg Ott
Helsinki University of Technology
jo@netlab.tkk.fi

Cheng Luo
Helsinki University of Technology
cluo@netlab.tkk.fi

## ABSTRACT

Traditional approaches for communication security do not work well in disruption- and delay-tolerant networks (DTNs). Recently, the use of identity-based cryptography (IBC) has been proposed as one way to help solve some of the DTN security issues. We analyze the applicability of IBC in this context and conclude that for authentication and integrity, IBC has no significant advantage over traditional cryptography, but it can indeed enable better ways of providing confidentiality. Additionally, we show a way of bootstrapping the needed security associations for IBC use from an existing authentication infrastructure.

Categories and Subject Descriptors: C.2.0 Computer-communication networks: Security and protection

General Terms: Design, Security

Keywords: Disruption- and delay-tolerant networking, identity-based cryptography, initializing security

## 1. INTRODUCTION

Many 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 immediately. 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) [1].

Due to the special nature of DTN environments the traditional security mechanisms are not always applicable [2]. For example, end-to-end confidentiality using traditional encryption mechanisms requires the sender to know a recipient-specific encryption key. If the sender does not already have this key, and has no immediate connectivity to the recipient or a supporting server, the sender will not be able to send a private message to the recipient.


*MobiOpp'07,* June 11, 2007, San Juan, Puerto Rico, USA.


In this paper we analyze the applicability of identity-based cryptography (IBC) for DTN security and conclude that IBC does not offer significant advantage for message authentication, but it helps in providing confidentiality. We also show that bootstrapping from an existing authentication infrastructure, such as the mobile cellular network, provides a good way to initialize the needed secure connections for IBC use.

## 2. RELATED WORK

The Delay Tolerant Networking Research Group (DTNRG)[1] is developing protocols for the communication of DTN messages, or "bundles". A draft DTNRG document presents the bundle security protocol specification [3] and an additional draft document [4] explains the rationale for the design choices made in the specification. The specification describes three security headers that can be added to bundles to provide different security services. The Bundle Authentication Header (BAH) is used to provide authentication over a 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 and the Confidentiality Header (CH) is used to encapsulate encrypted payload of a bundle. Different combinations of these three security headers can be used simultaneously.

Identity-based cryptography (IBC) [5] is a relatively new cryptographic method that enables message encryption and signature verification using the public identifier, such as e-mail 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 $PP$.

[1] `http://www.dtnrg.org`

Seth et al. [2] 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) [6] in which different regions have subregions 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.

# 3. APPLICABILITY OF IDENTITY-BASED CRYPTOGRAPHY

In this section we analyze the applicability of IBC for DTN communication and discuss how the needed security association between the PKG and a principal could be initialized.

## 3.1 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 predefined 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 [3] 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. [2] 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 15th, 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 traditional digital signatures are used for authentication, the sender can compute all the necessary authenticators even when there is no network connectivity.

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

To summarize, IBC and traditional digital signatures are equally effective as mechanisms to authenticate DTN messages. In both cases, the sender must have been able to receive a message (containing the IBC key or the certificate respectively) from a server (PKG, or CA respectively) sufficiently recently. The receiver can authenticate DTN messages even while disconnected.[2]

## 3.2 End-to-end confidentiality

In open networks, communication confidentiality is achieved by applying encryption to messages. 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 recipient-specific encryption key in order to encrypt the message for a certain recipient. Without connectivity (to recipient or supporting server) obtaining the key and checking its validity is difficult.

In IBC systems, a sender can encrypt 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 [2].

In [4] 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. Thus, with respect to the 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 behavior by using traditional cryptographic techniques in a way that is suitable to DTNs?

We answer this question by constructing two different designs for *private messaging* in *disconnected environments*

[2]If the sender is completely disconnected, it can still send a DTN message with a traditional digital signature. Recipients capable of fetching the current certificate for the sender can authenticate this signature.

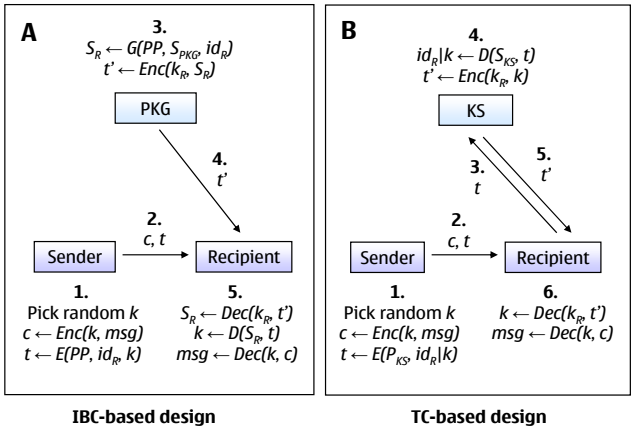


Figure 1: Private messaging in disconnected environments

(see Figure 1). 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 $k_R$.

### 3.2.1 IBC-based design for private messaging

In IBC-based design the server is an IBC private key generator PKG (that has public parameters $PP$ and the corresponding secret key $S_{PKG}$). $PP$ and the identity of the recipient $id_R$ are known to $S$ ahead of time. We denote asymmetric IBC encryption and decryption with $E()$ and $D()$, 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, $S$ picks a random symmetric key $k$, encrypts the message $msg$ using symmetric encryption and creates an envelope $t$ by encrypting $k$ using IBC encryption for identity $id_R$ (step 1). $S$ sends both the ciphertext $c$ and $t$ to $R$ (step 2). To decrypt the received ciphertext $c$, $R$ needs an IBC private key issued by the PKG for $id_R$. $R$ connects to PKG, which generates an IBC private key $S_R$ for $R$. This private key is encapsulated into a recipient envelope $t'$ using $k_R$ (step 3). PKG sends $t'$ to $R$ (step 4). $R$ decrypts the $t'$ 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 3.1, 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.

### 3.2.2 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 $S$ 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, $S$ picks a random symmetric key $k$ and computes a ciphertext $c$ using symmetric encryption. Then $S$ creates 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 $R$ (step 2). $R$ forwards the envelope $t$ to KS (step 3) which decrypts $t$ and constructs a recipient envelope $t'$ that holds $k$ (step 4). KS sends $t'$ to $R$ (step 5) which first recovers $k$ from $t'$ using $k_R$ 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.

### 3.2.3 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.

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 (1) it has less strict requirements on recipient's network connectivity and (2) it imposes considerably less load on the

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

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

[1]If $R$ sends several envelopes in step 3, KS can package all decryption keys belonging to $R$ into a single recipient envelope, reducing the number of symmetric key encryptions performed by KS.
[2]If KS packages multiple decryption keys into single envelope, the receiver has to do fewer decryptions.

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.

## 3.3 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 [7].

Seth et al. [2] 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. suggest that authorized distribution agents like kiosks could handle the enrollment. The kiosk maintainer should identify the new enrolling user and verify that the user is really entitled to use the e-mail address with which she is enrolling to the system.

Another approach would be to use some form of return routability check to verify the ownership of particular e-mail address. At the time of enrollment the PKG could e.g. send a secret enrollment key by e-mail to the claimed e-mail address. However, this approach 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.

If an existing security infrastructure is available a good alternative is to bootstrap the needed secure connections from there [7]. Bootstrapping removes the need for manual identification and verification checks (like the ones e.g. the kiosk maintainer would have to do) and it also provides better security compared to return routability checks.

The mobile cellular infrastructure that consists of credentials provisioned by cellular operators to cellular subscribers e.g. in the form of SIM cards, and roaming agreements among different cellular network operators is by far the most widely deployed authentication system, with more than two billion users and hundreds of participating cellular operators. The Generic Authentication Architecture (GAA) [8] provides a way 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*

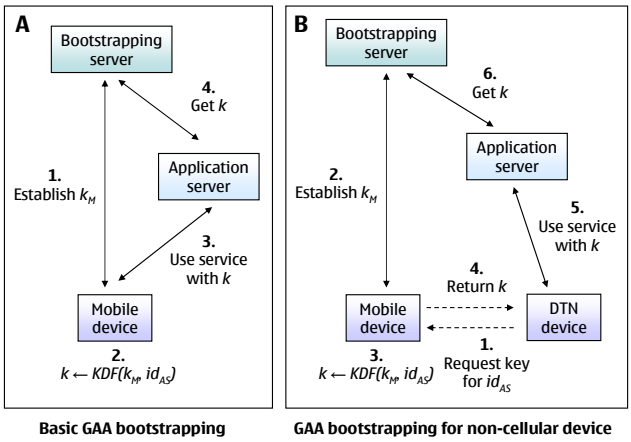


Figure 2: Initializing security with GAA

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 current GAA specifications require two-way interaction between the bootstrapping server and the mobile device (step 1). The next version is expected to support "push"-style bootstrapping which would allow (a) the communication in step 3 to be unidirectional from application server to mobile device only, and (b) this communication would also carry the information needed for bootstrapping, so that communication in step 1 between the mobile device and bootstrapping server would not be needed.

In DTN infrastructure CA and PKG servers can act as GAA application servers, run either by the cellular operator or independent third parties that have service agreements with the operator. The CA can use GAA to authenticate the enrollment of public keys, and issue short-lived subscriber certificates. The PKG can use GAA to encrypt IBC private keys for devices. A principal is identified by a well-known identifier (e.g., e-mail address or mobile phone number) which is securely bound to a cellular identifier. Re-

vocation of this identifier is automatically reflected in the DTN security infrastructure, since the device is no longer 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 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).

## 3.4 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. One way to overcome this limitation [2] is by using hierarchical identity based cryptography (HIBC) [6]. But using HIBC 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.

# 4. DISCUSSION

Based on our foregoing analysis of 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 could indeed lead to a more efficient solution for end-to-end confidentiality, in terms of load on the server as well as network connectivity requirements for recipients. Bootstrapping security associations from the cellular security infrastructure seems to hold promise in addressing the initialization, key management, and cross-realm operation aspects of DTN security.

We are currently building an implementation on laptops, tablets and mobile phones in order to experiment with the ideas presented in this paper. Our objective is to demonstrate that the concepts such as bootstrapping DTN security from the cellular authentication infrastructure work in practice.

Besides authentication and confidentiality there are also other important DTN security issues. As explained in Section 3.1 intermediate nodes should be able to decide whether to store and forward messages based on local state, such as remaining battery life, and the sender's identity. Forwarding policies could 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. We hope to address this issue of policy-based forwarding in our future work.

# 5. REFERENCES


[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. SIGCOMM*, August 2003.

[2] A. Seth, U. Hengartner, and S. 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`.

[3] S. Symington, S. Farrell, and H. Weiss, *Bundle Security Protocol Specification*. IRTF, DTN research group, October 2006. Draft version -02; expires in April 2007.

[4] S. Farrell, S. Symington, and H. Weiss, *Delay-Tolerant networking security overview*. IRTF, DTN research group, October 2006. Draft version -02; expires in April 2007.

[5] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *CRYPTO 2001, Advances in Cryptology*, no. 2139 in Lecture Notes in Computer Science, pp. 213–229, Springer-Verlag, August 2001.

[6] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," in *8th Intl. Conf. on the Theory and Application of Cryptology and Information Security*, December 2002.

[7] N. Asokan and L. Tarkkala, "Issues in initializing security," in *Proc. Fifth IEEE Intl. Symposium on Signal Processing and Information Technology*, pp. 460 – 465, IEEE Press, December 2005.

[8] P. Laitinen *et al.*, "Extending cellular authentication as a service," in *Proc. First IEE Intl. Conf. on Commercialising Technology and Innovation*, September 2005. `http://www-admin.iee.org/OnComms/PN/communications/062%20-%20P%20Ginzboorg.pdf`.