Identity-Based Cryptography for Delay-Tolerant Networking

Kevin Fall
Intel Research, Berkeley

Anirban Chakrabarthi
University of Iowa

May 23, 2004

Abstract

We present the security requirements for a Secure Delay Tolerant Networking Architecture and suggest Identity Based Cryptography (IBC) as a useful technique for meeting many of them. IBC compares favorably with traditional public-key cryptosystems by allowing the equivalent of public keys to be generated on-the-fly, as needed. This capability can, in principal, reduce the number of transactions required with a centralized trusted third party in addition to reduce overhead in messages (by possibly allowing the option for not carrying full public keys in messages). Such savings are especially important for a DTN, where network partitions may be frequent and lengthy.

1 Introduction

Practical identity based cryptography (IBC) has recently been introduced as a form of asymmetric cryptosystem sharing many of the characteristics of traditional public-key cryptosystems, but with some added benefits. With IBC, it is possible to encrypt and sign messages using a function of a public identification string of an intended recipient as the public key. These ID strings may be simple derivatives of the recipient's identity (e.g. e-mail address or combination e-mail address with time-stamp). By reducing the overhead and number of message exchanges required for operation, IBC simplifies public key management compared to traditional public key infrastructures, and thus appears to be very attractive for Delay Tolerant Networks (DTN), which by definition may be frequently partitioned. In this paper, we first review the background concepts of IBC and DTN, then outline a method for using IBC concepts to develop a secure DTN architecture.

2 DTN Security Requirements

The security requirements for a DTN include infrastructure protection (selective access to the function of message routing), authentication of routers and end-users, and optional confidentiality of message content. These capabilities are to be made available to security principals that might include individuals, organizations, and processes acting on behalf of either of these. DTN security has been discussed in an earlier white paper [10]. In the present work, we explore similar scenarios, but with an approach based on IBC instead of conventional (RSA-style) public key cryptography.

As with most distributed systems, the primary components of a secure DTN include authentication, access control and confidentiality:

- Authentication: Two uses of authentication are envisioned for use in a DTN. *Router authentication* allows DTN routers to establish authenticated channels between each other. In particular, a router receiving a route update message or traffic to be carried must be able to reliably establish the originator of the data. This use of authentication, among routers, is primarily used for ensuring accurate exchange of meta-data. *User authentication* reliably establishes identity of data originators. This use of authentication is to provide infrastructure protection (selected access to data forwarding or special quality-of-service features). Using IBC, we show how these two requirements can be met with a common scheme.

- Access Control: The access control facilities for a DTN relate to the ability of an end-user (or DTN host) to have its traffic carried on a DTN. Furthermore, access control may be employed to manage access to particular classes of service. For example, DTN routers within a region may only allow highest-priority traffic to be carried by principals identified as network operators within that region. Different portions of a DTN may provide different access control configurations, based on their specific requirements. As with most access control systems for distributed systems, cryptographically secure authentication is required to implement the access control scheme.

- Confidentiality: In some circumstances, an end-user agent may wish to make its data confidential and would like the DTN network software to implement this function. While we suggest confidentiality based on a particular IBC variant in this paper (Tate pairing on eliptic curves), it is clear that a general framework, able to accommodate a variety of enciphering algorithms, may be a requirement in certain specialized circumstances. In cases where a specialized algorithm is required by an application, the DTN-layer confidentiality mechanism might be bypassed in favor of encryption implemented at application layer.

Due to the performance characteristics of networks for which the DTN architecture was designed, there are certain design drivers which must be taken into account when designing the security mechanisms for the DTN architecture.

- Minimize message exchanges: A DTN may suffer frequent partition and may have long round-trip times. Message exchanges, especially client/server interactions, can therefore be burdensome on the network and may take a long time to complete. Thus, the DTN security interactions should try to minimize message exchanges between different entities as much as possible.

- Minimize dependencies on shared servers: A DTN system should try to minimize the access to any centralized server as much as possible, as there is no guarantee when any server will be reachable at any given point in time.

- Minimize message overhead: DTNs may exist in extreme environments where bandwidth is not plentiful. Thus, schemes which require smaller key material and key meta-data should be preferred to those

with larger needs (subject to ultimate security requirements).

- Facilitate access control revocation: DTN authentication and access control configuration databases may require updates as users are added or removed, or access rights are modified. A secure DTN scheme should provide some method by which identies and access control configurations can be modified over time.

These constraints pose a number of difficulties for several common security protocols that require multiple round-trip interactions to take place. For example, credential blacklisting may be difficult, as frequent, long-duration network partitioning will frustrate the system's ability to distribute blacklists quickly.

The security requirements mentioned above can be achieved using traditional public key cryptography, as suggested in [10]. The main problem with a scheme based on a traditional public key approach is that a sender has to obtain and verify the public key for every intended receiver. There are two ways this is typically achieved: (i) store the public keys for all potential communication peers in the sender ahead of time, or (ii) obtain any required public keys on-demand from a server which has the information. In a large system, the former approach is unlikely to be satisfactory, as the set of all possible peers may be large and difficult to predict. In a DTN environment, where connections may be intermittent, the latter is also not suitable. Furthermore, the overhead of requesting certificates or including copies of them with messages can be high (credentials are generally 512 - 1024 bytes long using RSA). This may introduce unacceptable overheads for DTNs with limited bandwidth carrying otherwise small messages.

3 Identity Based Cryptography

A common approach to developing secure distributed systems involves the creation of a Public Key Infrastructure (PKI) for managing, distributing and authenticating the keys used for public key cryptography. Verifying the authenticity of public keys is accomplished using *certificates* generated by a trusted Certification Authority (CA). Certificates are signed by the private key of the CA, thereby ensuring the correctness of the enclosed public key. Asymmetric cyphers such as RSA[] or ElGamal[] or some Elliptic Curve variant[] are frequently used. All users of the PKI are assumed to know the CA's public key in advance, and verifying the authenticity of an arbitrary public key thus requires obtaining and verifying the certificate for the key in question.

An Identity based Cryptosystem (IBC), originally introduced by Shamir [2] in 1984, simplifies some aspects of managing and distributing public keys.[1] In an IBC scheme, a sender can generate the public key of a potential receiver on-demand, using a pre-arranged simple (non-secret) string associated with the receiver (e.g. the receiver's e-mail address). This alleviates the need to certify the authenticity of public keys. A complication of the scheme, however, is that a receiver must be able to learn the private key corresponding to the public key from a trusted third party.

[1] It has a number of other useful properties. A high-level overview may be found in [**?**].

Since the introduction of the IBC problem two decades ago, no practical and secure solution had been developed until recently. In 2001, Boneh and Franklin at Stanford introduced a solution using a bilinearity property of pairings on elliptic curves (see next section).[2] Their approach, which we call the "BF scheme" has become the basis for both encryption and authentication using identity-based methods.

3.1 Identity Based Encryption

The BF scheme for IBC is based upon finding sets of points obeying *bilinear maps* on cyclic groups. Two such point sets are commonly used: the Weil and Tate pairings on elliptic curves. These pairings were known some ten years prior to their use in IBC by Boneh and Franklin. In particular, the Weil pairing was presented in[], and the Tate pairing discussed shortly thereafter[**?**].

To understand the approach we briefly introduce the study of elliptic curves over finite fields and then the use of pairings. The interested reader is referred to the excellent Pairings-Based Crypto Lounge web site[], and recent thesis by Maas[].

Elliptic curves arise in study of Diophantine equations, and had been studied traditionally over infinite fields (e.g. rationals or reals). Their study over finite fields did not generate much public interest until 1985 when Miller [] and Koblitz [] independently described how to use them in conjunction with discrete logarithms to form cryptosystems that might have better security than the conventional multiplicative group of a fininte field (or, equivalently, might be equal in security but with shorter keys).

A pairing on an elliptic curve is a function that takes two points as input and produces an element of some multiplicative Abelian group on output. A pairing is also understood to obey a non-degeneracy property and a bilinear property:

Let $(G_1, +)$ and $(G_2, *)$ be two cyclic groups of order q (q is some large prime). A map $e^{'} : G_1 \times G_1 \rightarrow G_2$ is called a *bilinear map* if $e^{'}(aP, bQ) = e^{'}(P, Q)^{ab}$ $\forall P, Q \in G_1, a, b \in Z_q$. We also require that $e^{'}(Q, R) = 1_{G_2} \forall R \in G_1 \iff Q = 0_{G_1}$ (non-degeneracy). From these, if P is a generator of G_1, then $e^{'}(P, P)$ is a generator of G_2.

The security of the system depends on the *Bilinear Diffie-Hellman Problem* (BDHP): given P, aP, bP, cP in $(G_1, +)$ compute $e^{'}(P, P)^{abc}$.

Under the BH scheme, the overall cryptosystem comprises four algorithms: **setup**, **extract**, **encrypt**, and **decrypt**. Setup generates global *system parameters* and a *master key*. Extract uses the master key to create private keys on-demand given an ID string. Encrypt encrypts a message using a public key ID, and decrypt decrypts a message using the corresponding private key. The system parameters must be known to all participants, and the master key must be kept secret from all participants (except from the party responsible for generating private keys, called the private key generator, or PKG).

It is to be noted, however, that deducing the master key from a private key is difficult: in particular, it

[2] Another scheme, based on quadratic residues, was introduced the same year by Cocks[1], but this scheme requires more overhead and will not be pursued further.

is hard to deduce s from P and Q, from the equation $P = sQ$, if both P and Q are points on the elliptic curve.

Computing a public encryption key at the sender involves knowing an appropriate receiver ID string and the system parameters. The situation at the receiver is more complicated: the appropriate matching private key is a function of the system parameters, the master key, and the chosen ID string. Because the master key is assumed to be kept secret, this in effect requires receivers to be assinged pre-generated keys or requires some transaction to be accomplished between a receiver and some trusted agent (i.e. a private key generator or PKG) to provide private keys to receivers as required. Note that encryption of this form at the sender has similar properties to tradional public-key cryptography: it provides confidentiality but not authentication.

An example of identity-based encryption is as follows: Alice sends a message to Bob encrypted with the string "bob@company.com+today's date". Alice requires only the system parameters to form the cyphertext. Bob has a corresponding private key either known a-priori, or is able to obtain such a key as required from a PKG. The advantage of this scheme is that if Bob is able to change his encryption key relatively frequently without requiring Alice (or any other sender) to perform a network transaction in order to obtain new key's for Bob, provided they know the method Bob is using to select ID strings . The process is illustrated in Figure 1.

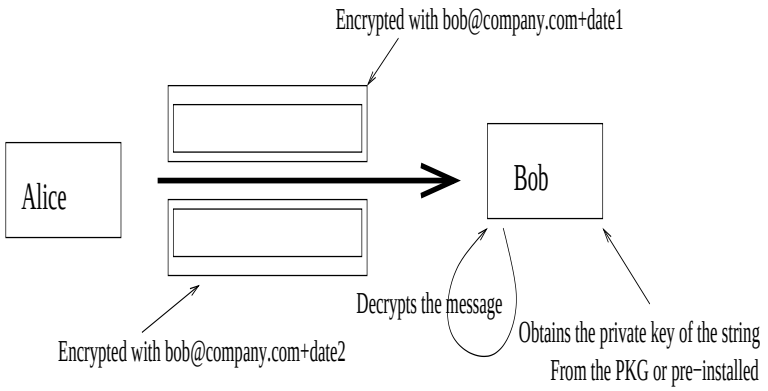

Figure 1: Obtaining the Initial System Parameters

3.2 Signatures and Access Control

Using the IBC techniques introduced by BH, which achieve privacy, several schemes been devised to create digital signatures One such technique, developed by Paterson [4], creates system parameters similar to the BH scheme. Unlike the BH scheme, however, the sender encrypts the message with his/her private key, and the receiver can authenticate the message knowing the ID string. Several other schemes have been proposed which develop ID-based signatures [5].

For capability-based systems, the access control matrix of principals and access rights is assumed to be maintained centrally. Once a capability is obtained, authorization is implied: the acess control decision is

made at the time of issuance. Furthermore, capabilities are like keys: they can be copied or provided to others by the holder. In an IBC-based system, the initial private keys loaded into the DTN devices may be used as capabilities. They are encoded in a straightforward fashion using IBC: the private keys (generated by the PKG) are identifiers that can be easily mapped to a vector of access rights where an object is accessed. Being able to encrypt a valid string with the appropriate public ID string implies the holder had access to the private key (and was therefore a capability holder).

Access control lists (ACLs) provide an alternative to this approach, in which the table of access rights may be maintained in a decentralized fashion, typically proximal with the object or service to be accessed. This approach relies heavily on verifiable authentication (via some credentials). The access control decision is made at time of access. Access rights granted by ACL entries are not transferable by the principal to others– the owner of the access list must make the modifications. In an IBC-based system, a pure ACL based approach is simple: the public ID strings of the principal are recorded with the corresponding access rights in access control lists.

Both capability-based and ACL-based systems have been implemented in local operating systems, where the access right enforcement is provided by a secured software layer or kernel. When extended to distributed systems, encryption is used to ensure capabilities or authentication material (required for using ACLs) may not be forged easily. This has lead to the creation of capabilities that are specialized for particular principal and enforced cryptographically (so-called *principal-specific capabilities*[**?**]).

With most cryptographic systems, it is standard practice to change key material regularly so as to reduce the probability that future data is compromised by having recovered key material from previously-captured transactions. Such practice extends to keys for principals in a distributed system, and with IBC this operation is especially easy: the public ID of the principal is appended with the date, as described above, to form the public ID string for IBC. Because the public ID strings may now change in a predictable way as a function of time (e.g. the date portion), any principal-specific capability or ACL entry would effectively change over time (because they contain public ID strings will likewise evolve over time). This places a (loose) requirement on time synchronization for the entire system.

3.3 Combining encryption and signing: *signcryption*

It is possible to combine both signing and encryption into a single operation (called "signcrypt"). Over the past few years, such schemes have been developed to allow a message to be decrypted and to have its signature checked at the same time. Combination of signature and encryption reduces the message size [6] and the total computational complexity. A study shows that around $30 - 40\%$ savings is possible by using signcryption.

3.4 Confidentiality of IBC system components

There are several important data structures required for operating an IBC system using the BH approach. The structures vary in their confidentiality requirements. While several structures are public (identity strings), others (e.g. the master key) must remain secret to avoid comprising the entire system. More specifically, the confidentiality requirements of the data structures are as follows:

Structure	Visibility	Description
ID String	Public	The string identifying the principal. May be generated on the fly by third pa
System Parameters	Public	Generated at system initialization time based on the master secret. All parti
Private Key	Private to principal	The private key corresponding to a particular public key. Must be generated
Master Secret	Private to authority	Key used to generate private keys and the system parameters. Public disclos

3.5 Consequences of IBC for DTN

The IBC approach does not offer a panecea for use in DTNs, but does provide what appears to be a superior factoring of responsibility between a sender and receiver as compared with more comon approaches. We now explore these differences in more detail.

IBC systems introduce the following advantages over traditional PKI systems:

- *Public Keys:* Public keys may be generated as required by a sender based on the system parameters and the ID string of the intended receiver. This avoids the the need to cache public keys for every possible receiver and related protocol and overhead to learn and verify the authenticity of public keys for new communication peers.
- *Key Changes:* Public/private key changes may be accomplished using a regular, published scheme for changing keys. For example, concatenating the date to the end of the ID string to form a time-sensitive ID string provides for a natural periodic way of changing keys without the associated overhead of having senders learn new keys.
- *Generating Symmetric Keys:* IBC can be used to generate a symmetric key without having to go through a message exchange (cf. Diffie Hellman).[7]
- *Reduced processing overhead:* IBC is based on computations on eliptic curves. Such computations are generally agreed to be considerably less complicated than exponentiation functions required by non-ECC type schemes (e.g. RSA).
- *Shorter Cyphertext:* Encrypion using IBC tends to produce shorter cyphertext and appears to require smaller keys for comparable security[]. This could lead to reduction of overhead in the xxx

In spite of the advantages, there are several potential disadvantages associated with use of an IBC system:

- *Single Point of Failure:* Public disclosure or compromise of the master secret compromises the entire system. This problem is partially addressed by secret splitting approaches which in effect spread the secret across multiple holders.

- *Distribution of Private Keys:* Private keys are generated with the master secret as input. Thus, for a network in which on-demand connectivity cannot be assumed, private keys may have to be pre-generated and loaded into a potential communications participant. If the private keys are exhausted, some mechanism must be invoked to update hosts with new keys.

- *IBC is relatively new:* IBC/BH is based on the difficulty of solving the bilinear Diffie Helman problem for eliptic curves. While believed to be hard, this topic is rather new for the public cryptography community (2001), and has received considerably less analysis than other methods.

4 A Secure DTN System Design

We propose a secure DTN system based on the IBC concept. We begin with defining the relevant system entities:

- *DTN Hosts:* Location where endpoint applications are present and a DTN agent is responsible for managing store and forward operations. The agent on a DTN host may communicate either with other DTN hosts directly or with DTN routers. Because of the characteristics of IBC, a host need not cache public keys for every possible next hop. DTN hosts must be capable of verifying the authenticity of messages on receipt.

- *DTN router:* Location where DTN message forwarding takes place, but no DTN applications are expected to be found. DTN routers have their own private keys and can decrypt or authenticate data directed for them.

- *DTN Private Key Initialization* This refers to the process by which a DTN router learns its private key(s), which could be accomplished by hand at creation time, or by some yet-to-be-specified key exchange protocol with a PKG. A PKG protocol may applicable for DTN environments where communication is comparatively frequent and low-delay.

4.1 IBC Parameters Setup

The first step in IBC setup is the generation of IBC system parameters using the BH method. First, a set of elliptic curve group parameters are derived. These consist of the choice of the curve itself and a generator point P. After the group parameters are selected, a master secret s is selected, which is generally a random value in a range specified by the group parameters. The master secret is required later to derive the private

key corresponding to any identity string id. [3] The generator and master secret are used to produce the system public key $P_{pub} = sP$. The resulting system parameters .... include the group parameters and the system public key. These must be available to any participant in the system.

4.2 DTN Host Initialization

Host initialization is achieved by learning the system parameters, any private keys for future communications, and any necessary credentials (see below).

In a DTN environment it is desirable to have the system parameters and the initial private keys of the hosts pre-installed in the DTN devices. This saves the devices from having to obtain the parameters after deployment from some PKG server. However, certain scenarios may arise where it may become necessary to obtain the system parameters from the PKG, for example, if the system parameters or master secret changes. In such cases, a straightforward secure channel can be established with a PKG server (see appendix **??**).

4.3 Access Control

In the secure DTN architecture, access control mechanisms are needed by routers to ensure arriving traffic is authorized to be carried over the DTN (and, if policy requires) what classes of service may be given or how much traffic may be carried. A similar requirement may apply to DTN hosts when they receive messages from other hosts or routers. Access control mechanisms are likely to be useful in administering access to various other objects in a DTN. Access control has traditionally been implemented by use of capabilities or access control lists.

4.4 Soft-State Access Control

Because updating capabilities or ACL entries in a DTN may be comparatively difficult due to periods of persistent disconnection, it may be desirable to augment capabilities or ACL entries with timers and/or counters to provide a certain form of *soft-state based access control*. In particular, timers/counters may be used to reduce (or increase) access control privileges or invalidate a capability automatically, in the absense of communications that explicitly makes such modifications. This is similar in spirit to soft-state protocols, which remove state if not updated in a certain amount of time. Here, however, we do not necessarily remove the access rights. Rather, we allow them to degrade (or increase) to some predefined level, which may not be all-or-nothing.

There are several options for establishing soft-state access control. Each has a set of advantages and disadvantages. In the following discussion, the term *token* is understood to mean either an ACL entry or a principal-specific capability:

[3] This works essentially by first converting the id to a point on the curve, and then calculating the private key as product of master key and the generated point on the curve.

- *Timer-based tokens:* These tokens expire after a certain amount of time unless refreshed (and may not be valid until a certain point in the future is reached). The appropriate timer value would need to be based on the risk associated with continued access after desired revocation versus the probability of being disconnected. If network communication was very infrequent, longer timer expirations would likely be required (and vice-versa). Other options are possible (e.g. rather than revoking access entirely upon expiration, access rights could degrade to some pre-specified system default). There is a possibility of unfair resource allocation with this scheme: a comparatively fast or well-connected principal may be able to complete a larger number of transactions during its validity interval than a less-well-connected competitor.

- *Counter-based tokens:* In this case, access is restricted to some finite number of operations unless refreshed (e.g. so many messages forwarded). It is effectively a credit scheme. This avoids the time synchronization issue implied by the timer-based based approach above, but requires persistent state to be maintained regarding the number of previous access attempts. It provides equal resource allocation, unlike the timer-based scheme, which could lead to underutilization.

- *Rate-based tokens:* This approach represents a hybrid of the two previous approaches, and limits the number of accesses per unit time.

Whether a token refers to an ACL entry or a PSC can have a drammatic effect on the overall system operation. For time-based capabilities, the validity time is encoded in the capability. This approach therefore requires time synchronization between the capability issuer and checker in order to establish the validity of the capability. For time-based ACLs, only a local notion of time is required: the local system decides about the validity of the access request without reference to any other clock. For counter-based capabilities, the count is encoded in the capability which must be reduced when accesses take place (a sort of debit card with the balance residing on the card). This presents a significant challenge, as it requires the older capability to be revoked and a new capability produced with a reduced count encoded inside. For counter-based ACL entries, the corresponding count must be persistently stored. Furthermore, if the counters are intended to be global (e.g. total of number of messages allowed to be sent), then the count must be at systemwide scope. This requires access control enforcement points to communicate among themselves to ensure the ACL entry counters are globally synchronized.

4.4.1 Host Access Control

In the host access control, the host needs to send credentials authorized by a trusted party to gain access to certain routers. Host access control can be time-based, sequence-based or count-based access control. In case of sequence based access control, the host requests the PKG for a certain sequence number interval (start and end sequence numbers), for which the PKG sends private keys stored by the host to be used when required. The request string can look like this, “hostname+start sequence+end sequence”. The host

decides to encrypt the message with the public key of the PKG. The PKG may decide to divide the sequence numbers into smaller sequence intervals and generate keys for each of these smaller intervals and send the set of keys to the host encrypted with the host's public key. It is to be noted that the host will be able to use the keys only when it is sending the bundle specific to the sequence interval. The host sends the credential with the bundle at the start of the sequence interval. The router caches the credential. When the sequence interval expires, the host need to send a new credential to the router.

For both time-based and count-based access control, the same principle is followed. In case of time-based access control, the host requests for credentials for a certain period of time, indicating start and end time. The PKG provides keys by dividing the time interval into time slices.

4.4.2 Router Access Control

Both time-based and count-based access control can be used for router access control. In case of count-based access control, the router gets credential keys (private keys corresponding to number of bundles and the next-hop router) from the PKG. The request can look like "hostname+nexthop router+bundle count". The PKG generates the keys it may decide to break the bundle count into smaller bundle intervals. For example, the router requests for 1000 bundles to the next hop router. The PKG generates keys for the following strings "hostname+nexthop router+100+i", where 100 indicates that the key is for the next 100 bundles and i indicates the key id (ranging from 0 to 9 in this case). Due to the presence of the key id the router would not be able to reuse the same key for different bundle intervals. The only difficulty with this kind of approach is that the router needs to generate credentials for different routers it is connected to. Therefore, it has to predict before-hand which router would be the most popular next hop for it. However, it can always generate credentials once it is out of credential keys for the next hop router, by requesting the PKG to generate credential keys for the router for a certain number of bundle intervals.

4.5 Updating the Private Keys

Another host-PKG interaction occurs when the PKG decides to refresh the system parameters and thus decides to update the private keys of each of the hosts in the DTN system. Updating of system parameters is a very infrequent event and may be triggered periodically. This should be as infrequent as possible, as there is a scalability problem for a large system. Host may also request the PKG for an update of system parameters, in case its private key is compromised. Ultimately it is up to the PKG to decide whether the refreshing of private key and system parameters are necessary. Otherwise, a host can be compromised and the mechanism can be used as a tool for Denial-of-Service attack. It is to be noted that for replicating PKGs, it is a good idea to update the system parameters are they are being replicated throughout the system and chances of the master key being compromised is very high, if the system parameters are not changed at all. On the other hand, for multiple PKGs, the global master key need not be updated as the system is more or less fixed for the global master key. However, each region may decide to update its own system parameters.

However, the process is more scalable compared to the replicated PKG, where the system parameters need to be updated for all systems within DTN.

When a PKG actually decides to generate a new set of system parameters, it broadcasts to all the nodes. A hash value of the packet is encrypted with its private key ensure the confidentiality of the broadcast. The PKG can then send the new private keys to the hosts, encrypted with their old private key, to maintain data confidentiality. It is upto the hosts to get new credentials based on the updated system parameters from the PKG.

4.6 Providing Data Confidentiality and Integrity

When a host sends a bundle to another host, the integrity and confidentiality of the bundle must be maintained. Integrity indicates that the data cannot be modified by anyone along the path and confidentiality indicates that the data cannot be read by anyone along the path. In addition, the host needs to present to the routers, the credentials necessary for access control. The integrity of the credentials also need to be checked. Also, it should be guaranteed that no one can actually stick a different credential to the bundle.

Data Confidentiality: To ensure data confidentiality, two levels of encryption schemes are employed. The host generates a random number, which is used as a symmetric key and the whole message is encrypted with the symmetric key. For the symmetric key encryption, AES scheme can be employed. The symmetric key is also encrypted with the public key of the receiver. It is to be noted that the public key of the receiver is a string. This encryption is done using the BH scheme. The two levels of encryptions is similar to the encryption employed in PGPMail. Instead of this type of two levels of encryption, the Sakai scheme can be used to generate a symmetric key. In that scheme, the sender generates the symmetric key using the public key of the receiver and its private key, and the receiver does the same. It can be shown that the symmetric key thus generated are equivalent.

Data integrity: To ensure data integrity, the host bundle is hashed and the has value is encrypted with the private key of the host. Scheme mentioned in [4] is used for encryption purposes. Any node, on receiving the message, can validate the authenticity of the message by using the public key of the host. Instead of separately providing confidentiality and integrity, several signcrypt schemes mentioned in literature, which simultaneously provide confidentiality and data integrity [9].

Credential Transfer: The host appends the credential for the particular interval (can be sequence, time or count interval), with the bundle at the start of the interval. It may also decide to maintain the integrity of the credential, by using the method described above. When a router receives a credential, it caches the credential. The router has generated similar credential and appends the credential to the bundle and forwards it to appropriate router or inter-region gateway.

4.7 Storage at the host

It is to be noted that, using the IBC system, the storage at the host is greatly reduced. The host needs to store the following:

- *System parameters:* These parameters are obtained from the PKG, and these include the elliptic curve, system public key P_{pub} (generally 512 bits), and a couple of hash functions.
- *Own private key:* This refers to the private key corresponding to the string "hostname", where hostname refers to the string which identifies the host in the DTN environment.
- *Credential Keys:* These are keys corresponding to the credentials that the host decides to generate. Generally, each key is of 512 bits long, so if the host stores 10 keys in advance then the total storage because of these keys amount to 5kb.

It is to be noted that the storage at the host is doubled if the system choose to implement Multiple PKGs instead of Replicating PKGs.

4.8 Dealing with Bundle Fragmentation

Bundle fragmentation is an important issue which needs to be taken into account during the design of secure DTN infrastructure. Secure DTN bundles have two components: end-to-end components and hop-by-hop components. The end-to-end components do not provide any extra challenge, because the packets will be reassembled and then the confidentiality and the integrity of the bundle will be checked. However, for the hop-by-hop component the fragmentation has to be dealt with. In DTN, fragmentation are of two types [13]. In the first type of fragmentation (Type I fragmentation), the host/router proactively fragements a bundles into multiple self-identifying smaller parts. In the second type of fragmentation (Type II fragmentation), the host/router reactively choose to fragment a bundle on receipt, which it does if a portion of the original bundle has arrived. In case of Type I fragmentation, the router replicates the same credentials to the different fragments. It is to be noted that each fragment will be treated as a bundle thus increasing the count of the bundles that the router is supposed to send. In the type II fragmentation, there are two cases. In case 1, the bundle carries the credential (first bundle in the time/sequence interval). In case 2, the bundle does not carry the credential (middle bundle in the time/sequence interval). In case 1, if the credential is lost, the bundle is given the lowest priority, and treated as it would treat the bundles asking for the lowest priority. If the credential is still intact, then the router knows what to do with it and acts accordingly and stores the credential. In case 2, the router already knows what to do with these bundles and hence acts accordingly.

5 Issues regarding the PKG

5.1 Implementation of PKG

As mentioned earlier, PKG remains an area of concern in the IBC infrastructure. The issue becomes more acute in a DTN environment as accessing the PKG to obtain private keys all the time is not feasible. There are two ways to implement the PKG: (i) Replicating PKGs and (ii) Multiple PKGs.

5.1.1 Replicating PKGs

In this type of implementation, the contents of the PKG is replicated throughout the DTN environment. The contents include the system parameters and the master key. The main advantage of this system is that it is simple to implement and the flexibility of the IBC design is not compromised. The main disadvantage of this implementation is that since the master key is replicated throughout the system, therefore the risk of the master key getting compromised by the adversary is increased manifold.

5.1.2 Multiple PKGs

This type of implementation is slightly complex than the previous one. In this type of implementation, there are two types of PKGs: the global PKG containing the key information of the whole DTN system and the local PKG containing the information of the different regions within the DTN system. The global PKG generates private keys for each of the hosts/routers in the system for the string "hostname", which is the identifying string for the host/router in the whole DTN system. Since the global DTN system is a fixed environment (public keys are fixed), therefore there is no need for each host to get the private key from the PKG and can be installed at the time of setup. Therefore, global PKGs do not exist in each region, and the private keys are stored in each host within the DTN system. The other PKG is the local PKG, which is responsible for generating keys within the region. These PKGs also will have a set of system parameters. These PKGs will be able to generate keys on demand depending on the encrypting string ID. The system parameters of the local PKGs extend only in the region and do not have any meaning outside the region. One disadvantage of this type of implementation is that, each host and router need to store both global and local system parameters, thus doubling the amount of storage required in case of replicating PKGs.

It is to be noted that subsequently, during the discussion of the protocols in detail we assume a replicating PKG. The protocols can be used in conjunction with multiple PKGs with simple modifications.

Obtaining the System Parameters: The first thing that a host needs to get from the PKG is the information about the current system parameters. The host requests to the PKG for the system parameters. The PKG sends back the system parameters. Since the information is public, there is no need to encrypt the above information. However, there is a need to prove that the information is actually coming from the PKG, and nobody else is indulging in a man-in-the-middle attack. To prove the authentication of the message, the PKG hashes the system parameters and encrypts them using its private key. Let S be the

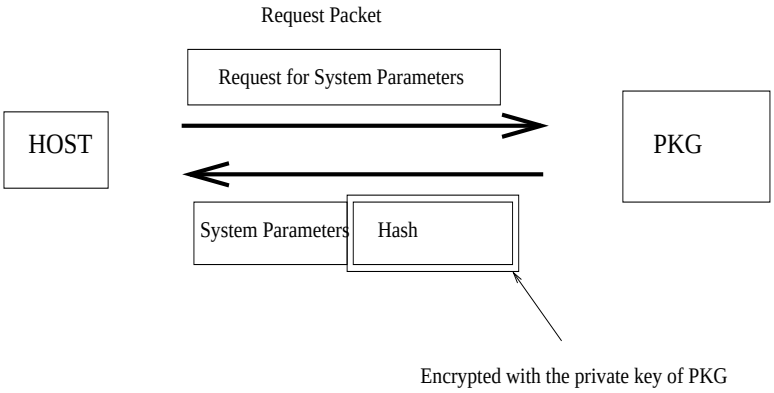

Figure 2: Obtaining the Initial System Parameters

set of system parameters, and H be the hash function. Then the PKG sends back the following message, $[S, E_{PKG}(H(S))]$. The host will be able to validate the authenticity of the information, by validating the hashed information with the obtained system parameters. It is to be noted that an adversary modifying the information along the way is only a nuisance as the host will be able to detect that the received system parameters are not accurate. If the host is able to authenticate the message, it caches the system parameters for future use. The process is illustrated in Figure 2.

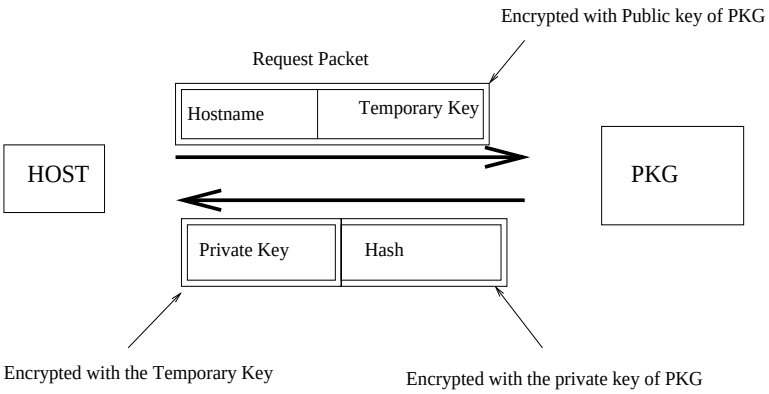

Figure 3: Obtaining the Initial Private Key

Obtaining the initial private key: Getting the initial private key is one of the most difficult problems. The reason is that the host needs to get the information in a secure manner, and since this is the first key the host is receiving, therefore the PKG cannot encrypt the key with a secret information known to only the two of them. The protocol to obtain the secret key is as follows: The host sends a request packet to the PKG, indicating that it needs to get the current private key for the string "hostname", which is its identifying string used throughout the system. The information is encrypted with the public key of the PKG, which is simply the address of the PKG. In addition, the host also provides a random number called the "temporary key", which is to be used for encrypting and sending the message back. The PKG generates the private key, and sends back the private key encrypted with the temporary key. In addition the PKG also computes a

hash of all the information and encrypts it with its own private key. This is used to provide integrity to the whole information, and the host can easily determine if anybody has changed the information in between. The host then stores the private key in case it needs to authenticate any message. The key is stored till it gets a new key from the PKG. The process is illustrated in Figure 3.

References

[1] C. Cocks, "An Identity-Based Encryption Scheme Based on Quadratic Residues," *Proc. 8th IMA Intl. Conf. on Cryptography and Coding*, LNCS2260, Springer-Verlag, 2001, pp. 360-363.

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. CRYPTO,* pp. 47-53, 1984.

[3] D. Boneh and M. Franklin, "identity Based Encryption from the Weil Pairing," in *SIAM Journal of Computing,* vol. 32, no. 3, pp. 586-615, 2003.

[4] D. Paterson, "ID based signatures from Pairings on Elliptic Curves," in *Electronic Letters,* vol. 38, no. 18, 1025-1026, 2002.

[5] J. Cha and J. Cheon, "An Identity-Based Signature from Gap Diffie Hellman," in *PKC 2003 LCNS 2567,* pp. 18-30, 2002.

[6] B. Lynn, "Authenticated Identity-based encryption," *available at* eprint.iacr.org.

[7] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on Pairings," in *Proc. SCIS 2000,* Jan. 2000.

[8] C. Gentry and A. Silverberg, "Hierarchical ID-based Cryptosystems," in *AsiaCrypt 2002,* 2002.

[9] J. Malone Lee, "Identity-based Signcryption," available at *http://eprint.iacr.org/2004/004.*

[10] , R. Durst, "An Infrastructure Security Model for Delay Tolerant Networks", July 2002, Available from http://www.dtnrg.org.

[11] , N.P. Smart, "Access Control Using Pairing-Based Cryptography," *topics in Cryptology; CT-RSA 2003, LNCS 2612*, Springer-Verlag (2003), pp. 111-121.

[12] , Martijn Maas, "Pairing-Based Cryptography," MS Thesis, Technische Universitat Eindhoven, Jan 2004.

[13] V. Cerf et. al., "Delay-Tolerant Network Architecture," *IRTF Draft,* draft-irtf-dtnrg-arch-01.txt, Oct. 2003.