Privacy and Confidentiality in Context-Based and Epidemic Forwarding

Abdullatif Shikfa*, Melek Önen, Refik Molva
EURECOM

Abstract

Autonomic and opportunistic communications require specific routing algorithms, like replication-based algorithms or context-based forwarding. In addition to confidentiality, privacy is a major concern for protocols which disseminate the context of their destination. In this paper, we focus on the confidentiality and privacy issue inherent to context-based protocols, in the framework of an original epidemic forwarding scheme, which uses context as a heuristic to limit the replication of messages. We define the achievable privacy level with respect to the trusted communities assumption, and the security implications. Indeed, privacy in such an environment raises challenging problems, which lead us to a solution based on refinements of two pairing-based encryption, namely searchable encryption and identity-based encryption. This new solution enables forwarding while preserving user privacy by allowing secure partial matches in the header and by enforcing payload confidentiality.

Key words:
privacy, confidentiality, bilinear pairings, opportunistic communication, context-based forwarding

*Tel. : +33 (0)4 93 00 82 41 - Fax : +33 (0)4 93 00 82 00
Email addresses: shikfa@eurecom.fr (Abdullatif Shikfa), onen@eurecom.fr (Melek Önen), molva@eurecom.fr (Refik Molva)
[1]EURECOM, 2229 route des crêtes , BP 193, F-06560 Sophia-Antipolis cedex, France

1. Introduction

Context-based forwarding (e.g.[1, 2, 3]) is a new communication paradigm, where messages are forwarded from source to destinations based on the context (e.g. location, workplace or social information) instead of explicit addresses. To be more precise, each message is associated with a message context which corresponds to the profile of its destination, and nodes make their forwarding decisions by comparing the context of the message with their own profile and the profile of their neighbors.

The assumption behind context-based forwarding is that the larger the context shared by two nodes, the higher the chances for these two nodes to meet one another (e.g. two persons working at the same company are highly likely to be in transmission range at some point). Thus, routing decisions in context-based forwarding are guided by the similarity between the contexts of encountered nodes and the destination's context.

Context-based forwarding is particularly adapted to challenged heterogeneous environments, like opportunistic and autonomic networks [4], where end-to-end connectivity is not guaranteed. Indeed, in such environments, classical routing mechanisms may be impractical, hence the transmission of messages should rely on opportunistic strategies.

Opportunistic networking consists in transmitting the messages over any communication medium available. Many opportunistic forwarding protocols are replication based (e.g. [5, 6, 7, 8, 9]). The replication factor depends on a heuristic which is used by intermediate nodes to decide either to forward the message or to drop it. Using the context as heuristic is interesting for controlled epidemic forwarding: a node decides to forward a message only if the shared context between the message and the node is significant.

However, such a protocol presents major security issues. Firstly, a classical security requirement is payload confidentiality: the payload should only be accessible by the legitimate destination and therefore requires end-to-end encryption. This issue calls for innovative solution in the context of opportunistic networks since such network are delay-tolerant and thus do not support end-to-end key agreement. Moreover, since the destination is defined implicitly through context information, even the identity of the destination itself is not necessarily known by the source.

Secondly, user privacy is a crucial issue in such a protocol. The message context is indeed essentially a subset of the profile of the destination, and the message is forwarded through various intermediate nodes that may not

be trusted by the destination or the source. Moreover, trust relationships are loose in such a heterogeneous environment, therefore nodes want to keep a tight control over the access by other nodes to their profile due to privacy reasons. Thus, nodes should be able to correctly make context-based forwarding decisions on encrypted messages. The encryption mechanism itself should be public and should not require prior contact with the destination, while avoiding dictionary attacks. To this end, we propose a new solution based on refinements of two pairing-based cryptosystems, namely Identity-Based Encryption [10] and Public Encryption with Keyword Search (PEKS) [11]. The tailored use of these functions with particular settings enables intermediate nodes to detect matching attributes without revealing the non-matching ones, and enforces confidentiality of the payload of the message, with no online access Trusted Third Party.

Related work in this area is scarce because context-based forwarding is an emerging concept and existing security solutions do not fit the issues presented above.

In [12], Lilien et al. present several challenges in privacy and security of opportunistic networks, and in particular the need for end-to-end confidentiality but they do not propose solution and they do not analyze the issue of context privacy and in particular the requirement of matching encrypted context.

In the neighboring area of Delay Tolerant Networks (DTN), the DTNRG defined a Bundle Security Protocol (BSP [13]) to secure communications in DTN. BSP includes in particular a confidentiality block that only enables the encryption of the entire payload at the source and its decryption at the final destination based on the identifier of the destination. The confidentiality capabilities offered by BSP do neither enable encryption based on the context of the destination nor partial matches computations that are required to take context-based forwarding decisions: those features are therefore not flexible enough to fit the requirements of context based forwarding.

Finally in [3], Nguyen et al. propose a context based forwarding protocol with a security solution based on public hash functions to enable partial matches at intermediate nodes while allegedly protecting privacy of the destination. Their approach is yet prone to dictionary attacks and does not achieve the claimed privacy as analyzed in section 3. To the best of our knowledge, we are thus the first to analyze the problems of privacy in context-based opportunistic networks and to propose a solution sketch in [14]. Compared with [14], this article includes several new developments:

the solution is improved with a new payload encryption mechanism for payload confidentiality, it includes an extended version with multiple TTPs, an analysis of the issue of revocation and a formal security evaluation.

The main contributions of this paper are as follows:

- We introduce a communication design that combines the concepts of epidemic and context-based forwarding,
- We study the problem of payload confidentiality and user privacy in this framework, and define the trusted communities assumption. We further define the security primitives required to achieve privacy in the proposed protocol.
- We propose an original design that features complete context-based and epidemic forwarding with strong confidentiality and privacy enforcement. End-to-end confidentiality is provided through an extension of Identity-based encryption where the identity is replaced by the sum of the attributes of the destination, and context-based forwarding decisions are taken while preserving user's privacy through a modification of a searchable encryption scheme with the help of an offline trusted third party.

In the next section, we first describe the context-based and epidemic network model and then focus on the security requirements of the introduced model. Section 3 analyzes a basic approach based on hashes, whereas section 4 presents our original scheme based on Identity-Based Encryption and Public Encryption with Keyword Search. In section 5 we evaluate our solution both from a security and performance point of view, and we conclude in section 6.

2. Problem statement

2.1. Context-based and epidemic forwarding

Classical routing mechanisms are not well adapted to opportunistic and autonomic networks, due to nodes' high mobility which implies unstable network topology and the lack of end-to-end connectivity which results in unpredictable end-to-end delays. In such environments, the transmission of messages relies on opportunistic strategies like epidemic forwarding. The main challenge in epidemic forwarding [7] is to find a suitable heuristic in

order for the nodes to decide either to carry and forward the message or to drop it and avoid network congestion.

We present an original design of an epidemic forwarding protocol which heuristic is based on the context (e.g. personal information, residence, work, hobbies, ...). The assumption behind context-based forwarding is that the larger the context shared by two nodes, the higher the chances for these two nodes to meet one another (e.g. two persons working at the same company are highly likely to be in transmission range at some point). This assumption leads to an interesting context-based heuristic for controlled epidemic forwarding: the idea is that a node would decide to store and forward a message only if the shared context between the message and the node is significant, and to drop it otherwise.

To be more precise, we consider a network composed of a set of n nodes $\{N_i\}_{1 \leq i \leq n}$. The context of a node N_i is defined as a set of attributes $\{A_{i,j}\}_{1 \leq j \leq m}$, where each attribute $A_{i,j}$ is a couple attribute name E_j, attribute value $V_{i,j}$. The set of attribute names $\{E_j\}_{1 \leq j \leq m}$ is known by all nodes. Finally, the profile $Prof(i)$ of node i is the concatenation of all its attributes: $Prof(i) = A_{i,1}||...||A_{i,m}$. All nodes have the same set of m attributes, but the value of an attribute may vary between nodes.

This model is illustrated by the small network given in figure 1. In this example there are $m = 3$ attributes ($E_1 = Mail$,$E_2 = Workplace$ and $E_3 = Status$) and $n = 4$ nodes (N_1,N_2,N_3 and N_4); each node's profile is an instantiation of the set of attributes (e.g. $Prof(1) = (Mail, alice@inria.fr)||(Workplace, INRIA)||(Status, student)$).

When a node N_S ($1 \leq S \leq n$) wants to send a message M to a destination N_D ($1 \leq D \leq n$), N_S divides M in header $\mathcal{H}(M)$ and payload $\mathcal{P}(M)$, $M = \mathcal{H}(M)||\mathcal{P}(M)$. The header $\mathcal{H}(M)$ holds the profile of N_D known by N_S:

$$\mathcal{H}(M) = ||_{j \in L} A_{D,j}$$

where $L \subset [1, m]$ is the subset of the indexes of the values of $Prof(D)$ that N_S knows, and $||_{j \in L} A_{D,j}$ denotes the concatenation of the attributes with such indexes. When an intermediate node N_i receives the message M, N_i compares its own profile $Prof(i)$ with the header $\mathcal{H}(M)$ to extract the subset $Q \subset L$ of indexes of values that are shared between $\mathcal{H}(M)$ and $Prof(i)$, such that $\forall j \in Q, A_{D,j} = A_{i,j}$. After this subset extraction, N_i can estimate its probability of meeting the destination, called matching ratio, as $p_i(M) = |Q|/|L|$, where $|X|$ denotes the cardinal of a set $|X|$.

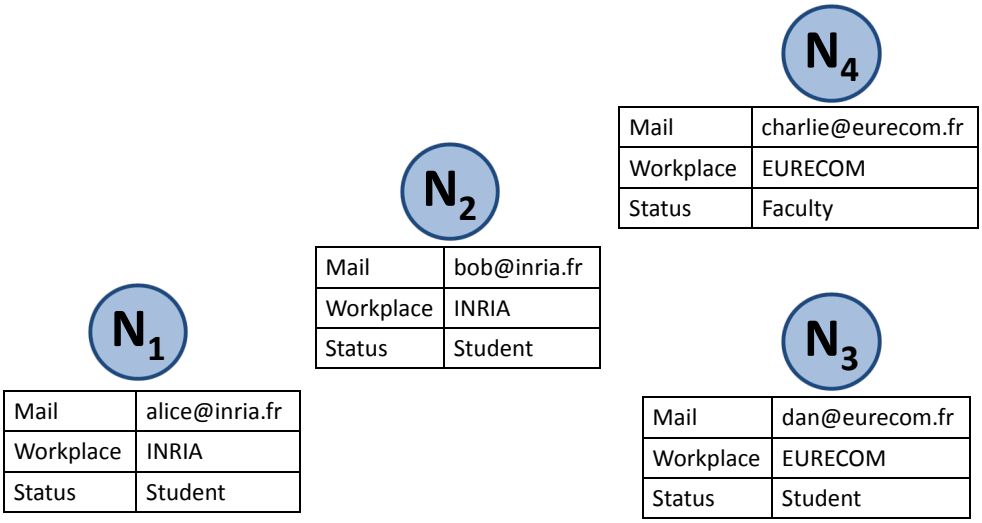

Figure 1: Simple network used as illustration. The network is composed of four nodes, each one has a profile composed of three attributes represented below the node. We consider two scenarios in this example: in the scenario (I), N_1 sends the message M_1 for all the students, and in the scenario (II) N_4 sends the message M_2 to N_1.

The matching ratio $p_i(M)$ is used as a metric and N_i decides either to drop the message or to carry and forward it depending on $p_i(M)$. The choice of the precise heuristic based on this metric impacts the performance of the protocol, and in particular on the number of replicas of a message in the network (which can be further limited through aging mechanisms like Time To Live) but the heuristic does not impact on the protocol security which is the focus of this article. The selection of the best heuristic is therefore out of the scope of this article but, for illustration purposes only, we choose a simple heuristic which consists for a node N_i in picking a uniformly distributed random number $0 \leq r_i \leq 1$ and then carrying and forwarding a message M with probability $p_i(M)$ (if $r_i \leq p_i(M)$) and dropping it with probability $1 - p_i(M)$ (if $r_i > p_i(M)$).

In this protocol, destinations of a message are implicitly defined as being all nodes which profile corresponds to the header of the message, therefore a message can have multiple destinations: a node N_i knows that it is a destination of M if its profile includes all the attributes in the header of the message resulting in $p_i(M) = 1$ (complete match). In the sequel of the paper, destination thus refers to one node or to a set of nodes depending

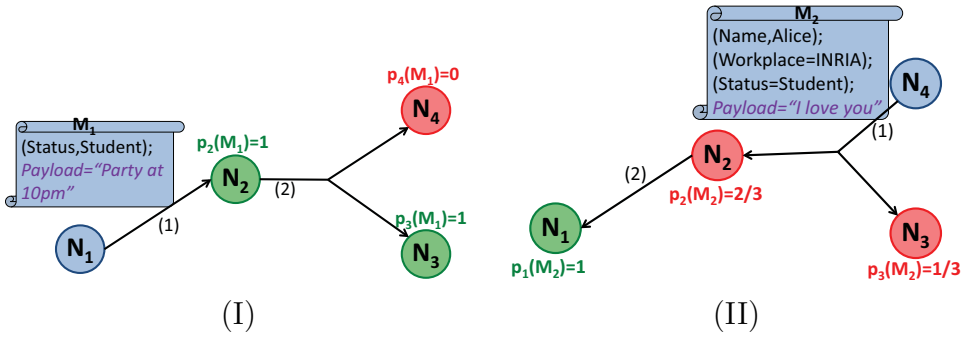

Figure 2: Two communication scenarios. In scenario (I), N_1 sends a message M_1 to all students (multiple destination) and in scenario (II) N_4 sends a message M_2 to N_1 (single destination). The nodes in blue are the source of messages, the nodes in green the destinations, and the node in red intermediate nodes that are not destinations.

on the header. To illustrate this fact let us consider two scenarios in the example of figure 2.

In the first scenario, we assume that node N_1 wants to send a message to all students to advertise a party in the evening. N_1 broadcasts the message M_1 with the payload $\mathcal{P}(M_1) =$ "$Party\ tonight\ at\ 10pm$" and a simple header composed of only one attribute: $\mathcal{H}(M_1) = (Status, student)$. The message is received by N_2 which computes $p_2(M_1) = 1$; therefore N_2 is a destination of M_1. The message is then broadcasted and received by N_3 and N_4 which compute their respective matching ratio as $p_3(M_1) = 1$ and $p_4(M_1) = 0$; thus N_3 is a destination of M_1 whereas N_4 is not.

As a second scenario, let us consider that N_4 wants to send a love declaration to N_1. N_4 broadcasts the message M_2 with payload $\mathcal{P}(M_2) =$ "$I\ love\ you$" and a complete header

$$\mathcal{H}(M_2) = (Mail, alice@inria.fr)||(Workplace, INRIA)||(Status, student).$$

The message is received by N_2 and N_3 which respectively compute the matching ratio $p_2(M_2) = 2/3$ and $p_3(M_2) = 1/3$. This indicates that neither N_2 nor N_3 are destinations of M_2, but N_2 is more likely to carry and forward M_2 than N_3. Finally when the message reaches N_1, N_1 computes the matching ratio as $p_1(M_2) = 1$ and concludes that it is a destination of M_2 (the unique destination since the mail address is a unique identifier). In the sequel of the paper, we use these scenarios to explain the security issues in such a forwarding protocol and refer to them respectively as scenario (I) and scenario (II).

The introduced protocol takes forwarding decisions based on the destination context, which is potentially sensitive information. This protocol hence needs to be enhanced with security mechanisms from a confidentiality, privacy and robustness perspective. To this end, we first define the security requirements in the next section.

2.2. Security requirements

As for classical communication mechanisms, data confidentiality is one of the first security requirements that should be taken into account. Indeed, access to the content of any message should only be authorized to destined nodes. Data confidentiality is usually ensured by using cryptographic encryption algorithms. However, contrary to many existing solutions based on symmetric cryptosystems, context-based communication protocols cannot rely on an end-to-end key management mechanism. Therefore source nodes should be able to encrypt the content of the message without a priori sharing any key with the destination node(s). Moreover, since the identity of destination nodes may be unknown by the source (e.g. in scenario (I)), the key used for the new encryption mechanism should be based on the set of attributes included in the context of the message. Only nodes which own the correct set of attributes should be able to access the content of the message. There is thus a very strong link between decryption keys and attributes used to encrypt the message. Only authorized nodes, i.e. those which really own the corresponding attributes, should receive the required decryption keys; the decryption keys should therefore also depend on the set of attributes included in the context of the message.

The encryption mechanism should also be flexible in the sense that a node should be able to encrypt a message with any set of attributes but only nodes owning all attributes that were used to encrypt the message should be able to decrypt this message using the combination of keys corresponding to each of the attributes. For example, in scenario (II), the encryption key should be derived from the three attributes in the header $((Mail, alice@inria.fr)$,$(Workplace, INRIA)$ and $(Status, student))$ and only N_1 should have the keys to decrypt the payload, but N_4 should not need to agree on an end-to-end key with N_1 beforehand.

Hence, in order to ensure payload confidentiality and only let authorized nodes access the payload, the two following security primitives have to be formally defined:

- ENCRYPT_PAYLOAD: used by the source to encrypt the payload of the message for the destination based on its attributes. This function should be public and the encryption key should be based on attributes.
- DECRYPT_PAYLOAD: used by the destination node to decrypt the encrypted payload of the message. This function should be private: since messages are encrypted with respect to attributes, only nodes who actually have those attributes should receive the decryption keying material.

Furthermore, as opposed to classical forwarding or routing algorithms, context-based forwarding algorithms allow nodes to take forwarding decisions based on the context information instead of a specific address. Since the context uses information on the user's characteristics and therefore is very sensitive, such protocols raise new privacy concerns which conflict with the communication protocol. Indeed, the context included in the header of the message should be secret for privacy reasons. Yet, while being kept secret from any other node, the message still needs to reach the correct destination nodes. Therefore, forwarding nodes should be able to compare their profile to the context included in the header of the message without having access to unshared context information.

Thus, the security requirements differ from those in payload confidentiality: while payload confidentiality is an end-to-end service where only nodes with a complete match (with matching ratio equal to one) can decrypt, the dedicated encryption function should still enable nodes with partial matches (with a matching ratio less than one) to discover shared attributes and hence to correctly forward packets.

Therefore, full destination nodes' privacy cannot be assured and we define a weaker model for privacy that we call *trusted communities* assumption. Communities are defined on attributes' basis: all nodes sharing a given attribute A_j form the community of attribute A_j. In the example of figure 1, nodes N_1, N_2 and N_3 form the community of students. In the trusted communities assumption, nodes which belong to the same community trust each other and do not harm each other. Hence, revealing to another node a shared attribute is acceptable from a privacy perspective, but attributes that do not match should remain secret.

The trusted communities assumption makes it also easier to manage colluding attackers. For instance, suppose that an encrypted message is

sent to faculty in INRIA. A node N_1 with attributes $(Status, Student)||$ $(workplace, INRIA)$ may collude with another node N_2 with attributes $(Status, Faculty)||(workplace, EURECOM)$ and obtain the key needed to decrypt the message. Similarly, they could decrypt messages sent to students in EURECOM. This attack implies that N_1 and N_2 merge their decryption capabilities and are able to impersonate each other. In particular, it means that N_2 now can discover the attribute $(workplace, INRIA)$ in any message and harm the INRIA community even though N_2 works at EURECOM. Thus, by colluding with N_2, N_1 would violate the trusted communities assumption (and similarly for N_2). Collusion attack is therefore ruled out by the trusted communities assumption.

We therefore consider that nodes might adopt a malicious behavior for a specific message only based on the knowledge they get from the message and if they discover an attribute that do not match their profile. For instance, they could decide to drop all messages addressed to a certain profile in a Denial of Service (DoS) attempt against nodes with the said profile. The protection of the header is therefore important not only for privacy but also from a robustness perspective. For example, in scenario (II), node N_2 could decide to drop the message M_2 if it discovers that the $Mail$ of the destination is $alice@inria.fr$, but N_2 correctly executes the protocol if it only discovers the $Workplace$ and $Status$ attributes which it shares. Attacks which do not depend on the knowledge gained from a message (e.g. where a node randomly drops messages) are out of the scope of this paper. Such attacks are indeed mainly due to selfish nodes, and can be mitigated through soft security mechanisms like cooperation enforcement schemes ([15, 16, 17]).

Moreover, as for payload confidentiality, encryption of the header cannot rely on an end-to-end key management mechanism because of the opportunistic nature of the communication medium. In fact, any node, should be able to send a secret message to any other node even if it does not share a single attribute with that one. Therefore, the encryption function should be public.

To sum up, in order to ensure user's privacy, context-based forwarding protocols require the definition of the following two additional security primitives:

- ENCRYPT_HEADER: used by the source to encrypt the context information. This function should be public and should enable forwarding nodes to compare their profile with the encrypted context in order to

correctly forward packets.

- MATCH_HEADER: used by any forwarding node to determine whether the encrypted header includes some shared attributes of its profile. This function should not however reveal any additional information on non-matching attributes.

The primitive ENCRYPT_HEADER denotes a secure encoding of the header and is different from traditional encryption schemes, like the payload encryption for example, in that decryption is not required. However, we use the term "encrypt" in both cases for the sake of clarity.

To summarize, a source node N_S uses ENCRYPT_PAYLOAD and ENCRYPT_HEADER to encrypt respectively the payload and the header of the message. Whenever an intermediate node receives an encrypted message, it first uses MATCH_HEADER in order to securely compute the matching ratio. If there is a complete match, i.e. if the matching ratio is one, then this node is a destination of the message and it performs DECRYPT_PAYLOAD on it. If there is a partial match, i.e. if the matching ratio is less than one then the node takes a forwarding decision depending on the matching ratio and the chosen heuristic.

Finally, since there is a strong link between each node's profile and the keying material used both for payload confidentiality and user privacy, nodes should prove that they really own the claimed attributes to at least one trusted entity and receive the corresponding keying material from this entity. Therefore, in the sequel of the paper, we assume the existence of a Trusted Third Party (TTP) which is in charge of verifying that nodes' profile are correct and of distributing the related keying materials if so. However, the correct execution of the forwarding protocol should not rely on the presence of this TTP: because of the delay-tolerant nature of the network, the TTP only plays a role on the preliminary distribution and possibly updates of the keying material and is considered offline during the forwarding of the data.

We now examine a first approach to meet these security requirements.

3. Basic approach based on hash functions

3.1. Solution sketch

The first idea to solve the privacy issue is to use hash functions, as proposed in [3]. A cryptographic hash function *hash* is an efficient one-way

function that does not require the use of a secret key and is preimage resistant: given h, it is difficult to find any M such that $hash(M) = h$.

The idea would be to use a hash function $hash$ to implement the ENCRYPT_HEADER primitive. To be more precise, a node N_S, which wants to send a message M to N_D, would simply hash all the values of the header $\mathcal{H}(M)$, thus obtaining:

$$\mathcal{H}(M) = ||_{j \in L}(E_j, hash(V_{D,j})).$$

As a counterpart, an intermediate node N_i would implement MATCH_HEADER as follows:

- N_i would first hash its profile with the same hash function $hash$,
- N_i then tests whether one of its hashed attributes $(E_j, hash(V_{i,j}))$ is equal to an attribute $(E_j, hash(V_{D,j}))$ of the received header.

The preimage resistance property of $hash$ implies that the attributes where the equality holds are shared attributes, and the one where it does not hold are non-matching attributes.

This idea seems attractive because it requires only a public function, which is $hash$. Furthermore, hash functions are widely available, and they are efficient to compute.

3.2. Dictionary attack

The idea of using hash functions does not meet the security requirements defined in section 2.2. This solution is prone to dictionary attacks and such attacks have a strong impact on the privacy of the solution. Indeed, in context based communications, attributes are not pseudo-random sequences, they are rather well formated and have a meaning. Therefore, the intermediate node can compute the hash of each word in a dictionary and then simply look up the values of the message header in the hashed dictionary to discover the values of all (matching and non-matching) attributes of the header. Dictionary here means all the possible values for a given attribute. For example, the attribute *Status* in network 1 can only take three values in a university: $student$ or $faculty$ or $staff$.

Since the hash function is public, dictionary attacks can easily and efficiently be launched by any node. Therefore hash functions, as they are used here, do not provide confidentiality or privacy.

In order to avoid dictionary attacks, two important properties are required:

- MATCH_HEADER, the counterpart of ENCRYPT_HEADER, should be private. This property means that a node should only be able to match the values of attributes in its profile and no other values. This was already mentioned in section 2.2, but the basic solution does not verify this property. This property also implies that ENCRYPT_HEADER and MATCH_HEADER should be different functions.

- The output of ENCRYPT_HEADER should be randomized, which means that the output of ENCRYPT_HEADER should be different at each execution, even if the input does not change. All nodes need to be able to compute the function ENCRYPT_HEADER on any input in order to be able to send a message to any destination, therefore ENCRYPT_HEADER has to be implemented by a public function as mentioned in section 2.2. If the output of ENCRYP_HEADER is deterministic, nodes can launch a dictionary attack on all possible inputs as explained above, but this attack cannot be launched if the output of ENCRYPT_HEADER is randomized.

We now present our original security design which fulfills these properties.

4. Our proposed solution

4.1. General idea

Our goal is to provide payload confidentiality and user privacy by encrypting the payload and parts of the header while allowing the forwarding of messages. This implies that nodes with shared attributes are able to match these attributes in the header (partial match) and that only the destination can decrypt the payload (complete match).

The basic approach based on hash functions described in section 3 emphasized the importance of thwarting dictionary attacks by adding some randomness or salting. Salting was originally proposed in the context of password protection, to protect weak user passwords from pre-computation attacks or rainbow tables [18]. Yet, our problem is different because the communication protocol requires public encryption functions that can be used by any node without the need of a secret, and therefore cannot rely on shared keys which are required by symmetric techniques (e.g. salting).

Public encryption functions call for asymmetric cryptosystems. However, the implementation of classical schemes like RSA [19], requires the existence

of a public key infrastructure in order for a node to prove its identity: a node would fetch a destination certificate before sending a message. Such solutions are unfortunately not practical in an opportunistic environment.

Identity-based cryptography is a good candidate for opportunistic environments since it avoids the use of certificates while being asymmetric. Therefore, we propose a solution based on refinements of identity-based cryptography to allow any node to compute an encrypted version of the message.

The scheme is characterized by:

1. All nodes have the same set of m attributes, but the value of an attribute may vary between nodes. Nodes also get private keys corresponding to their attributes' values from an offline Trusted Third Party (TTP).
2. User's Privacy:
   (a) Each message contains $|L| \leq m$ (attribute, value) pairs, where the values are encrypted with Public key Encryption with Keyword Search (PEKS) functions.
   (b) A node can match its (attribute, value) pairs to that in the message header, again using PEKS functions and its private keys.
   (c) The forwarding of a message by an intermediate node is probabilistic; it is based on the number of matched attributes.
   (d) The main idea to provide the matching capability to intermediate nodes is to modify the instantiation of PEKS, by introducing an offline TTP.
3. Payload confidentiality:
   (a) The payload of each message is encrypted using ID-based Encryption (IBE) function.
   (b) A node can decrypt the message payload using ID-based Decryption (IBD) function, only if it matches all (attribute, value) pairs.
   (c) The main idea in implementing these end-to-end confidentiality functions is to use a sum of $|L|$ arguments instead of a single argument in IBD and IBE.

In the next section, we present an overview of the cryptographical background and then we focus on the innovative way in which cryptographic primitives are integrated in our scheme to meet the payload confidentiality and user privacy requirements.

4.2. Cryptographical background

4.2.1. Bilinear pairings and general settings

The cryptographic primitives used in this paper both for payload confidentiality and user privacy rely on cryptographic bilinear maps, therefore we briefly define them in this section. We consider two groups G_1 and G_2 of same large prime order q. G_1 is denoted additively and G_2 multiplicatively. We denote by P a generator of G_1. A cryptographic bilinear map is a function $e : G_1 \times G_1 \rightarrow G_2$ which satisfies the following properties:

- Bilinearity:

$$\forall P_1, P_2, P_3 \in G_1,\ e(P_1 + P_2, P_3) = e(P_1, P_3)e(P_2, P_3)$$
$$\text{and } e(P_1, P_2 + P_3) = e(P_1, P_2)e(P_1, P_3),$$

  By corollary, bilinearity also implies the following useful property :

$$\forall P_1, P_2 \in G_1, k \in \mathbb{Z}_q^*, e(kP_1, P_2) = e(P_1, P_2)^k = e(P_1, kP_2) \qquad (1)$$

- Non-degeneracy: $e(P, P) \neq 1$, which means that $e(P, P)$ is a generator of G_2,
- Computability: There exists an efficient algorithm to compute $e(P_1, P_2)$ for all $P_1, P_2 \in G_1$.

The security of cryptosystems based on pairings relies on the hardness of mainly two well-known problems:

- the Discrete Logarithm Problem which consists for a challenger in computing a given $< P, aP >$ in G_1 or in computing r given $< g, g^r >$ in G_2,
- the Bilinear Diffie-Hellman Problem which consists for a challenger in computing $e(P, P)^{abc}$ given $< P, aP, bP, cP >$.

q, G_1, G_2, e and P are global parameters of the system and are also used in the next sections.

4.2.2. Identity-Based Encryption

ID-based cryptography is a type of asymmetric key cryptography in which the public key of a node is the node's identity. ID-based cryptography allows a node to encrypt and send a message without previously receiving the destination node's public key. The first practical ID-based encryption scheme is proposed by Boneh et al. in [10]. In this section we remind the main components of their scheme in order to use them in the following sections.

Their scheme is based on a bilinear pairing e and therefore the node's identity which is used as a public key and as input for e has first to be mapped in G_1. A cryptographic hash function $H_1 : \{0,1\}^* \rightarrow G_1$ allows this mapping. In addition to all the parameters described in 4.2.1, the cryptosystem requires the existence of a Trusted Third Party (TTP) to generate nodes' private keys. To this extent, TTP first chooses a random number in $\mathbb{Z}_q^*$ denoted by TTP_{priv} which is kept secret, and publishes the public key as $TTP_{pub} = TTP_{priv} \cdot P$.

The encryption of a message M with identity ID outputs a ciphertext M' as follows:

$$M' = IBE(H_1(ID), M).$$

IBE is a randomized algorithm and therefore the M' changes at each execution even if the inputs are the same. To decrypt M' the destination needs first to fetch its private key from the TTP. The private key ID_{priv} corresponding to ID is indeed $TTP_{priv} \cdot H_1(ID)$ and can only be computed by the TTP. The decryption of the ciphertext M' is then performed with the Identity-Based Decryption primitive IBD as follows:

$$M = IBD(ID_{priv}, M').$$

In summary, identity-based cryptography allows a node to encrypt a message with only the knowledge of the destination's identity, and enables the destination to decrypt the messages after retrieving its private key from a TTP. The encryption mechanism is public and randomized, while the decryption mechanism is private (it requires the private key of the destination).

Identity based encryption is therefore a good candidate in order to achieve payload confidentiality in our protocol. However, since a source node may not know the destination identity in advance, and since the destination is defined as a set of attributes (in the header of the message), our protocol requires a new version of identity based encryption that takes a set of attributes as input. We therefore propose in section 4.3.1 an enhancement of identity-based encryption in a multiple attribute setting in order to provide encryption based on the conjunction of attributes instead of a single identity.

4.2.3. Public Encryption with Keyword Search (PEKS)

Another interesting construction based on bilinear pairings is Public Encryption with Keyword Search, introduced by Boneh et al. in [11]. PEKS allows a node to check whether some keyword exists in some encrypted data without being able to retrieve any additional information on the data. Moreover, this test can be performed only if the node has has previously received a trapdoor corresponding to the keyword. Thus, only authorized nodes can perform the test and they cannot learn any information apart from the occurrence or not of the keyword in the encrypted data.

To be more precise, a Public Encryption with Keyword Search scheme is composed of three primitives:

- $PEKS$ which takes as input the public key of a node A and a keyword W, and outputs W' a searchable encryption of W. $PEKS$ is a public and randomized function, and it is impossible to guess W given W' alone.

- $Trapdoor$ which takes as input the private key of a node A and a keyword W, and outputs a trapdoor TD for W. The $Trapdoor$ function is private because it requires the private key of node A, hence it is computable by node A only.

- $Test$ which takes as input an encrypted keyword W' and a trapdoor TD, and returns true if TD is the trapdoor corresponding to W and W' the encryption of the same W and false otherwise. In any case, $Test$ does not leak information about the actual value of W.

Figure 3 illustrates PEKS in a classical scenario with three nodes. The actual constructions of $PEKS$, $Trapdoor$ and $Test$ are given in B.

While PEKS seems a good candidate for user's privacy, this scheme cannot be used as is since, in the current version of PEKS, the sender needs to retrieve the public key of the destination in order to compute $PEKS$. Furthermore, $Test$ requires a trapdoor TD that can only be computed by the destination; the destination should therefore compute the trapdoors for its profile and send it to all the nodes and this is impractical in a DTN scenario. We, therefore, propose in section 4.3.2 to modify the instantiation of PEKS and to add an offline TTP to meet the requirements of header encryption in our protocol.

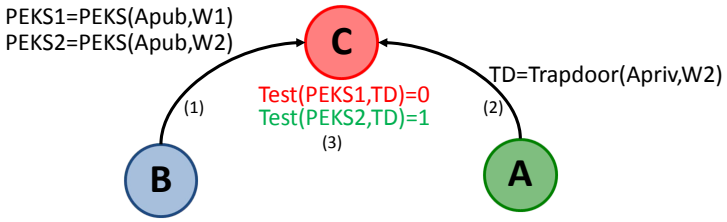

Figure 3: Functional description of PEKS. B is sending two PEKS values corresponding to W_1 and W_2. A gives C the trapdoor corresponding to W_2. C can then test the PEKS values received with the trapdoor and detects that $PEKS2$ corresponds to the trapdoor whereas $PEKS1$ does not.

4.3. Description of the scheme

We now focus on the description of our solution, and more specifically on the particular use of ID-based encryption and PEKS.

4.3.1. Payload Confidentiality

Payload confidentiality requires an end-to-end encryption mechanism between source and destination. The destination of a message is defined as a node whose profile contains all the attributes included in the header of the message and the encryption key should therefore also be derived from these attributes. As explained in section 4.2.2, we propose to use identity-based encryption in an enhanced multiple attribute setting that allows the encryption and decryption of a message with keys that are defined based on the conjunction of several attributes instead of a single identity.

The identity-based encryption mechanism ([10]) introduced in section 4.2.2 is based on bilinear pairings, and is therefore homomorphic with respect to addition. Thus, we propose to implement the payload encryption and decryption primitives as particular instantiations of IBE and IBD where the encryption and decryption keys become sums of several keys.

The encryption of the payload $\mathcal{P}(M)$ of the message M with header $\mathcal{H}(M) = ||_{j\in L} A_{D,j}$ (see section 2.1) is an IBE encryption with the sum of all attributes of the destination as identity:

$$\mathcal{P}(M') = ENCRYPT\_PAYLOAD(M) = IBE(\sum_{j\in L} H_1(A_{D,j}), \mathcal{P}(M)).$$

The destination profile includes by definition $A_{D,j}$ for $j \in L$, and therefore the destination has the corresponding private keys: $A_{priv_{D,j}}$. The destination

can thus decrypt the encrypted payload in message M' by using the sum of these decryption keys in IBD:

$$DECRYPT\_PAYLOAD(M') = IBD(\sum_{j \in L} A_{priv_{D,j}}, \mathcal{P}(M')).$$

The proof of correctness of the scheme is given in appendix A, while its security is analyzed in section 5.1.

This solution therefore provides end-to-end confidentiality between source and destination without the need to establish an end-to-end key or a group key beforehand.

4.3.2. User's Privacy

As described in section 2.2, while the protection of user's privacy requires the encryption of the header, intermediate nodes should still be able to compare their profile to the context of the message in order to correctly execute the protocol. PEKS enables intermediate nodes to perform searches on encrypted data without accessing the data. However, as explained in section 4.2.3, the sender requires the public key of the destination which is not practical in delay-tolerant networks. Furthermore, intermediate nodes would require to get the trapdoors from the destination which would defeat the purpose of protecting the destination's privacy.

We therefore propose to adapt PEKS functions to the environment defined in 2.1 in a way inspired by identity-based encryption. The latter alleviates the need for the destination's public key required in classical asymmetric schemes by replacing the public key of the destination by the public key of a TTP and the identity of the destination. Similarly, we propose to use PEKS with the public key of a Trusted Third Party instead of the public key of the destination. As a consequence, neither the public key of the destination nor the destination's identity itself need to be known and privacy is preserved. An example of this new instantiation of PEKS is sketched in figure 4.

To be more precise, the encryption of the header $\mathcal{H}(M) = ||_{j \in L} A_{D,j}$ consists in computing the PEKS value of each attribute in the header with the public key of the TTP resulting in the encrypted header $H(M')$ as follows:

$$\begin{aligned} \mathcal{H}(M') &= ENCRYPT\_HEADER(M) \\ &= ||_{j \in L}(E_j, PEKS(TTP_{pub}, A_{D,j})). \end{aligned}$$

This function is a randomized public function since $PEKS$ internally uses a random number.

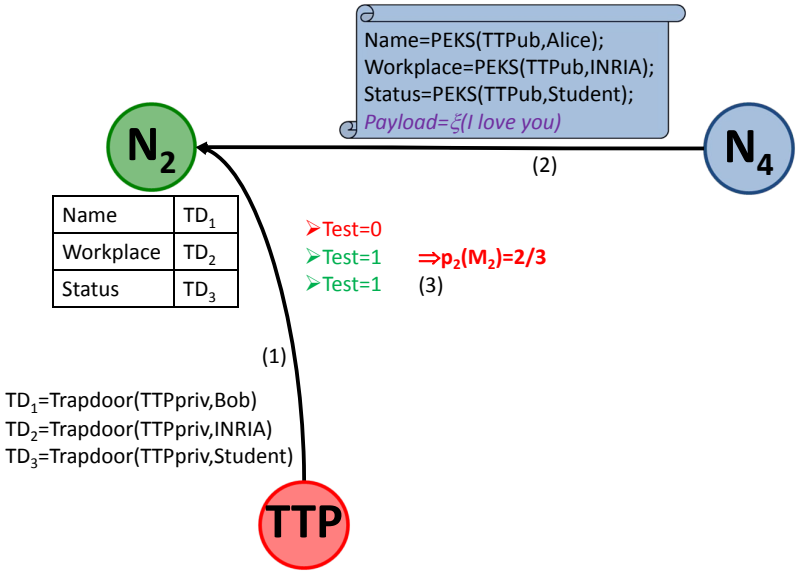

Figure 4: Example of user's privacy mechanism in scenario (II). The TTP gives nodes (here N_2) the trapdoors associated with their profile. The sender N_4 computes PEKS values of the header based on the public key of the TTP. Intermediate nodes like N_2 can then test the encrypted header with the trapdoors they own to compute the matching ratio $p_2(M_2')$. ξ denotes the PAYLOAD_ENCRYPTION primitive.

In order for an intermediate node N_i to compute the matching ratio, N_i uses the $Test$ function on the attributes of the header. To use this $Test$ function N_i requires trapdoors corresponding to its profile. These trapdoors can only be computed by the TTP since they require the private key of the TTP. Hence, N_i should receive the trapdoors associated with its profile which are $Trapdoor(TTP_{priv}, A_{i,j})$ for $1 \leq j \leq m$. Then, N_i implements MATCH_HEADER by extracting $PEKS(TTP_{pub}, A_{D,j})$ for each $j \in L$ from $H(M')$ and computing

$$Test(PEKS(TTP_{pub}, A_{D,j}), Trapdoor(TTP_{priv}, A_{i,j})),$$

which outputs:

- true, if $V_{D,j} = V_{i,j}$,
- false, if $V_{D,j} \neq V_{i,j}$.

N_i is therefore able to compute the matching ratio while the privacy of the destination is preserved. Indeed, even though the $Test$ function is public,

it requires the trapdoors as input, and each node only has the trapdoors corresponding to its own profile and cannot compute other trapdoors, because the *trapdoor* function requires the private key TTP_{priv} of the TTP. Hence, each node can only match attributes with its own profile as required in section 3.2.

4.3.3. Key Management

In the previous section, we presented security primitives achieving payload confidentiality and user privacy. These primitives require solutions for key management, and in particular a Trusted Third Party (TTP). However, because of the delay-tolerant nature of the network, the TTP should only have the role of distributing keying materials. It is considered as being offline during the execution of the communication protocol. We, therefore, define a preliminary step in the protocol where nodes can reach the TTP in order to receive their keying material.

During this setup phase, the TTP first generates public parameters of the system q, G_1, G_2, P, e, H_1 (see section 4.2). It also chooses a master secret TTP_{priv} and the associated public key $TTP_{pub} = TTP_{priv} \cdot P$. All nodes receive the public parameters and TTP_{pub}.

Furthermore each node N_i securely sends its profile $Prof(i)$ to TTP which verifies the validity of the said profile (the method of verification is out of the scope of this paper) and returns the private keys $\{A_{priv_{i,j}}\}_{1 \leq j \leq m}$ associated with $Prof(i)$. These private keys are required for the payload decryption, and the corresponding public key are simply $\{H_1(A_{i,j})\}_{1 \leq j \leq m}$.

N_i also requires the trapdoors as inputs for the $Test$ function and therefore N_i receives the set of couples $\{(E_j, Trapdoor(TTP_{priv}, A_{i,j}))\}_{1 \leq j \leq m}$.

When looking at the details of the implementation of $Trapdoor$ (see appendix B), it appears that

$$Trapdoor(TTP_{priv}, A_{i,j}) = TTP_{priv} \cdot H_1(A_{i,j}) = A_{priv_{i,j}}.$$

Hence, the legitimate trapdoors are the same as the legitimate private keys, which is interesting both from a security and performance point of view.

Each node N_i is required to enter setup phase before being deployed and being able to communicate with its peers, but this does not imply that nodes need to synchronize to enter setup phase at the same time. Before joining the network, node N_i needs to enter setup phase by contacting the TTP (e.g. through legacy networks like the internet). During this setup phase, N_i

receives public parameters q,G_1,G_2,P,e,H_1 and TTP_{pub} and N_i has m secrets $TTP_{priv} \cdot H_1(A_{i,j})$ for $1 \leq j \leq m$. N_i can then communicate with all nodes that already performed the setup phase, and does not need to contact the TTP anymore.

5. Evaluation

In this section, we evaluate the security and the performance of the scheme.

5.1. Security

We analyze first the security of the payload confidentiality scheme and then we focus on user's privacy.

On the one hand, the solution that ensures payload confidentiality is an extension of identity-based encryption in a multiple attribute setting.

In the original IBE scheme ([10]), the encryption key of a node with identity ID is $H_1(ID)$ and the associated decryption key is $ID_{priv} = TTP_{priv}H_1(ID)$. In our proposal we define the encryption key of the concatenation of several attributes $||_{j \in L} A_{D,j}$ as the sum of the hashes of each attribute $\sum_{j \in L} H_1(A_{D,j})$ and the corresponding decryption key is computed as:

$$\sum_{j \in L} A_{priv_{D,j}} = \sum_{j \in L} TTP_{priv}H_1(A_{D,j}) = TTP_{priv} \sum_{j \in L} H_1(A_{D,j}).$$

In [10], Boneh et al. proved that the basic construction of identity-based encryption is semantically secure against a chosen plaintext attack (IND-ID-CPA). The security proof is provided in the random oracle model and uses the widely accepted assumption that the Bilinear Diffie-Hellman problem is intractable.

The goal of an attacker Eve N_E in our proposal is to find the decryption key associated with attributes that she does not own. N_E might have some of the required private keys (because N_E shares some attributes with the destination) but not all of them: N_E misses at least one of the private keys required. We can separate the set L of indexes of attributes in the following partition: the set L_1 of indexes of attributes owned by N_E and the set L_2 of indexes of attributes that are not shared by the attacker, such that $L_1 \cup L_2 =$

L, $L_1 \cap L_2 = \oslash$ and $L_2 \neq \oslash$. The challenge for N_E is to determine

$$\sum_{j \in L} TTP_{priv} H_1(A_{D,j}) = \underbrace{\sum_{j \in L_1} TTP_{priv} H_1(A_{D,j})}_{known} + \underbrace{\sum_{j \in L_2} TTP_{priv} H_1(A_{D,j})}_{unknown}.$$

Thus the challenge for an attacker who knows some of the private keys is still an instantiation of the identity-based encryption in a multiple attribute setting but with less attributes. We show that our scheme is secure for any number of attributes and that an attacker cannot derive the private key of a message as long as there is at least one attribute whose private key is unknown to the attacker.

The IBE proof can be extended to the multiple attribute setting for this purpose and we consider only the set L_2 of indexes of decryption keys unknown to an attacker. The only difference is that the use of the addition operation implies a risk of collisions and therefore the risk of a possible decryption without the corresponding private keys.

To be more precise, suppose that there exists a set L_3 of indexes of attributes known by N_E such that

$$\sum_{j \in L_3} H_1(A_{E,j}) = \sum_{j \in L_2} H_1(A_{D,j}).$$

Then a direct consequence is that:

$$\sum_{j \in L_3} TTP_{priv} H_1(A_{E,j}) = \sum_{j \in L_2} TTP_{priv} H_1(A_{D,j}),$$

$$\sum_{j \in L_3} A_{priv_{E,j}} = \sum_{j \in L_2} A_{priv_{D,j}}.$$

Therefore if N_E manages to find a collision on the public keys with attributes that she owns and whose private keys are known to N_E, then N_E can decrypt the message without knowing the private keys $A_{priv_{D,j}}$ of the original attributes. We prove that the probability of finding such a collision is negligible.

The key point here is that the sums are operated on hashes of attributes and not on attributes themselves, and therefore the attacker has no control over the points in G_1 that he owns. The function H_1 is indeed a cryptographic hash function and in the random oracle model the output of the hash function

is considered to be uniformly distributed. If we consider a point $P_1 \in G_1$ then the probability for any input $A_{i,j}$ that $H_1(A_{i,j}) = P_1$ is $1/|G_1|$. And the same goes for the sum of random values in G_1: the probability that $\sum_{j \in L_3} H_1(A_{E,j}) = P_1$ is $1/|G_1|$. This holds true for any point $P_1 \in G_1$, and in particular if we consider $P_1 = \sum_{j \in L_2} A_{priv_{D,j}}$, this implies that:

$$\mathcal{P}(\sum_{j \in L_3} H_1(A_{E,j}) = \sum_{j \in L_2} H_1(A_{D,j})) = 1/|G_1|.$$

Such a probability is negligible (it is the same probability as finding an ID' different from a given ID such that $H_1(ID') = H_1(ID)$) as it amounts to the success rate of brute force attack.

Thanks to the use of the hash function H_1 collision is thus not an issue and Boneh's proof of semantic security of IBE applies in our setting as well. In particular it means that our proposal is also semantically secure against a chosen plaintext attack (IND-ID-CPA) and that a node can decrypt an encrypted message only if it knows all the required private keys and therefore is the destination. Other nodes cannot deduce any information about the payload even if some of the attributes are shared with destination nodes.

Finally we point at another interesting property of our scheme which is a consequence of the privacy preserving mechanism. As opposed to Boneh's scheme where the identity of the destination is sent in clear and accessible by any malicious node, the encryption keys can only be discovered by nodes which share the corresponding attributes. This property offers an additional security property as an attacker would first need to find which attributes were used to encrypt the message before being able to launch an attack as in the Boneh's attacker model.

On the other hand, user's privacy is preserved through a scheme derived from Public Encryption with Keyword Search. In [11], Boneh et al. proved that their construction is semantically secure against a chosen keyword attack in the random oracle model, assuming that the Bilinear Diffie-Hellman problem is hard. This means in particular that it is unfeasible for a node to discover the content of a keyword unless it knows the corresponding trapdoor. The computation of these trapdoors requires the private key of the TTP, and therefore only the TTP can compute and distribute them. Since the TTP provides a node N_i only with the trapdoors corresponding to its profile, this implies that N_i can only discover if some attributes in his profile are included in the header by using the $Test$ function: privacy is thus not

absolute, but this follows the trusted communities assumption described in section 2.2.

From an operational point of view, ENCRYPT_HEADER and ENCRYPT_PAYLOAD use only public functions and public keys that are distributed during the setup phase to all nodes. Any source node N_S can therefore send messages, even before meeting the destinations during the runtime phase. These primitives also avoid the dictionary attack, because they make use of internal randomization: their output is different at each execution, even if the input do not change. The proposed framework ensures privacy of the destination and confidentiality of the payload against eavesdroppers but also curious intermediate nodes, while enabling the computation of the probability used in forwarding decisions. Finally, the scheme features packets unlinkability and is therefore even resilient against traffic analysis from outsiders. Indeed, thanks to the randomness in ENCRYPT_HEADER and ENCRYPT_PAYLOAD, it is hard to link the headers of packets and therefore it is impossible to know if two packets have the same characteristics (in terms of destination or attributes) or not. Furthermore it is impossible to detect and analyze a communication flow because the forwarding decisions are taken probabilistically, therefore the route between a source and destination differs for consecutive packets.

5.2. Revocation

From a management perspective, the TTP provides all keys during the setup phase but it does not play any role in the runtime phase. This offline TTP is therefore compatible with an opportunistic network. Yet, as in many DTN protocols, key revocation is a difficult problem.

The problem of key revocation is a new problem that arises in the particular configuration in which we use PEKS but it was not an issue in the original PEKS scheme of Boneh et al. [11]. In [11], the revocation of the capability of using the $Trapdoor$ function was directly linked with the revocation of the private key of the destination and was therefore a classical problem. In our design, the issuer of the trapdoors is the TTP, and the same trapdoor is given to all nodes with the same profile. Yet profiles are dynamic and therefore it is important to be able to distribute new trapdoors to nodes which profile changed. In that case, trapdoors of other nodes (which profile did not change) should also be updated in order to guarantee a property like forward secrecy.

Concerning Identity-Based Encryption, Boneh et al. [10] suggest to solve the problem of revocation by adding a timestamp to identities. Instead of using simply the identity ID of a node as public key, one would therefore use (ID, t) where t is a timestamp. The TTP gives the node with identity ID the private key corresponding to (ID, t) at time t. The private key is automatically revocated after a period of time, because the public key that is used becomes $(ID, t + 1)$.

In order to come up with a solution to key revocation in the context of delay tolerant networks, we propose to divide the time in epochs θ^l, where l is a positive integer. For each epoch θ^l, the TTP generates a new private/public key pair TTP^l_{priv}/TTP^l_{pub} with $TTP^l_{pub} = TTP^l_{priv} \cdot P$, and nodes need to contact the TTP once during each epoch to get their updated keys. The use of epochs allows for a very loose synchronization between nodes and is therefore suitable for delay-tolerant networks.

The epoch's duration is chosen according to the network parameters such that all nodes can access the TTP once during an epoch. The duration of an epoch is therefore also considered longer than the time required by any packet to reach any node in the network. During epoch θ^l nodes need to enter setup phase with the TTP once to fetch the secrets corresponding to TTP^l_{priv}/TTP^l_{pub}, but they use the secrets of epoch θ^{l-1} to encrypt the messages because some nodes might not have fetched their secrets of epoch θ^l yet. Nodes also need to store the secrets of epoch θ^{l-2}: indeed during epoch θ^{l-1}, nodes use the secrets corresponding to θ^{l-2} to encrypt the messages. It is therefore possible that a message was sent at the end of epoch θ^{l-1} encrypted with the secrets of θ^{l-2} and is in the network at epoch θ^l. Since the duration of an epoch is longer than the time required by any packet to reach any node in the network, packets encrypted with older secrets than those of θ^{l-2} are automatically destroyed or dropped.

To summarize, nodes have a very loose synchronization since they only need to enter setup phase once in each epoch θ^l. The amount of secrets that they need to store is three times the amount of secrets required for one epoch; they indeed need to store the secrets corresponding to:

- θ^l once they fetch them,
- θ^{l-1} to encrypt the messages such that they can be decrypted by nodes that have not yet fetched the secrets of θ^l,
- θ^{l-2} to be able to decrypt the messages sent during epoch θ^{l-1} and that

have not expired yet.

The use of epochs therefore enables "delay-tolerant" key revocation over three epochs, while being compatible with the principles of delay-tolerant communication.

5.3. Protection against malicious TTP

The TTP plays a crucial role in the proposed framework. The TTP is indeed the only entity which can compute trapdoors and private keys of nodes based on the attribute values. The TTP is a trusted entity and therefore is assumed to behave properly but it is also a single point of failure that has the capability to decrypt all messages by using its private key TTP_{priv}.

It is therefore important to distribute the capabilities of the TTP and to remove the single point of failure. This is a new issue with respect to the original PEKS architecture. Indeed, while in the original design of Boneh et al. [11], the destination is computing the trapdoors and therefore there is no problem of key escrow, in our scheme the TTP is computing all trapdoors for other nodes. To this extent, we propose to distribute the trust and the capabilities on several third parties by adding the contribution of each one as follows. Assume there are w parties denoted by TTP_k with $1 \leq k \leq w$. All these entities use the same global parameters but each one generates a different private/public key pair denoted $TTP_{k,priv}/TTP_{k,pub}$ with $TTP_{k,pub} = TTP_{k,priv}P$.

Then, a source node N_S encrypts the header $\mathcal{H}(M)$ by using the sum of all the public keys of the TTPs as public key:

$$H'(M) = ||_{j \in L}(E_j, PEKS(\sum_{k=1}^{w} TTP_{k,pub}, A_{D,j})).$$

During setup phase, nodes N_i need to fetch trapdoors $(E_j, Trapdoor(TTP_{k,priv}, A_{i,j}))$ generated by each TTP_k. The total trapdoor associated with each attribute is the sum $\sum_{k=1}^{w} Trapdoor(TTP_{k,priv}, A_{i,j})$ of the trapdoors fetched at each TTP_k. These trapdoors can then be used as second input of the $Test$ function to compute the matching ratio. The proof of consistency of this scheme is based on the bilinearity of the pairings and is provided in B.

Concerning the security of the scheme, some TTPs might collude to retrieve the private key of another TTP. This attack is similar to the collision

attack described in the multiple attribute setting. Assume that the first $w-1$ TTPs are malicious and collude and that they want to discover the private key of the last TTP.

The goal of the colluding TTPs is to choose their public keys such that

$$\sum_{k=1}^{w-1} TTP_{k,pub} = TTP_{w,pub},$$

which automatically leads to

$$\sum_{k=1}^{w-1} TTP_{k,priv} = TTP_{w,priv}.$$

The problem though is that if the malicious TTPs choose their public key first, they cannot compute the associated private keys (because of the hardness of the discrete logarithm problem in G_1) and then they cannot derive the private key of the remaining TTP. Furthermore, a malicious TTP cannot fake a private key and provide nodes with fake trapdoors (by fake trapdoors we mean trapdoors that do not match their counterpart PEKS with the corresponding public key). Indeed, each node N_i receiveing a trapdoor $Trapdoor(TTP_{k,priv}, A_{i,j})$ from TTP_k directly verifies the validity of this single trapdoor by generating a $PEKS(TTP_{k,pub}, A_{i,j})$ and verifying that

$$Test(PEKS(TTP_{k,pub}, A_{i,j}), Trapdoor(TTP_{k,priv}, A_{i,j})) = 1.$$

Each share of the global trapdoor can thus be verified upon receipt, which prevents malicious TTPs from producing fake trapdoors. The remaining option for the malicious TTPs is to find $TTP_{k,priv}$ for $1 \leq k \leq w'$ such that

$$\sum_{k=1}^{w-1} TTP_{k,priv} P = TTP_{w,priv} P$$

which can be written as:

$$(\sum_{k=1}^{w-1} TTP_{k,priv}) P = TTP_{w,pub}.$$

It is clear that solving this equality amounts to being able to compute a discrete logarithm, while obtaining such a collision at random occurs with

probability $1/|G_1|$ as explained in the security evaluation of Identity-based encryption in the multiple attribute setting (section 5.1). Therefore, if up to $w-1$ TTPs are malicious and collude, they cannot create a collision or find the missing trapdoor of a given keyword, and therefore they cannot discover the content of encrypted keywords. If all w TTP collude though, it becomes as if the scheme consists of just one TTP, and the problems explained at the beginning of this section arise.

The same methodology can be used to solve the problem of distributing the trust over multiple TTPs for the payload encryption: the sum of the keys of the various TTPs is used in the encryption and the decryption process. To be more precise, the encryption of the payload of message M uses the sum $\sum_{k=1}^{w} TTP_{k,pub}$ of the public keys of all TTPs as parameter, and the decryption at the destination N_D of $\mathcal{P}(M')$ is performed with the sum of private keys of all TTPs for all attributes:

$$\mathcal{P}(M) = IBD(\sum_{j \in L} \sum_{k=1}^{w} A_{k,priv_{D,j}}, \mathcal{P}(M')).$$

The proof of correctness is based on the bilinearity of the pairing and is very similar to the one presented in appendix A and is therefore omitted.

As a conclusion, in order to alleviate the trust on a single entity, we propose to distribute the security capabilities (private key and trapdoor computation) among several third parties. In this new setting, nodes still use the functions defined in section 4.3.1 and 4.3.2 but apply the following simple modification:

- $TTP_{pub} = \sum_{k=1}^{w} TTP_{k,pub}$,
- $A_{priv_{i,j}} = \sum_{k=1}^{w} A_{k,priv_{i,j}}$,
- $Trapdoor(TTP_{priv}, A_{i,j}) = \sum_{k=1}^{w} Trapdoor(TTP_{k,priv}, A_{i,j})$.

The difference is simply that nodes need to contact several TTPs to get their secrets (but then they only need to store the sum of all these secrets so this does not incur an additional cost in terms of storage) and that no single TTP can break user's privacy or confidentiality; only the collusion of all w TTPs can result in such exposure.

5.4. Storage and Performance

Concerning storage, as mentioned in section 4.3.3, each node has to store m secrets in addition to the public parameters. Each of these secrets is in fact an element of a group of points on an elliptic curve of prime order p. The level of security in such groups depends on a security parameter called the MOV degree [21]: by carefully choosing the elliptic curve it is therefore possible to adjust the trade-off between key size and computation time, while maintaining a given level of security. In the case of mobile devices, computation resources are more constrained than storage, therefore it is preferable to choose a curve with small MOV degree, e.g. two. In such settings it is sufficient to have p of 512 bits length to have a security equivalent to 1024 bits RSA. The storage overhead of the secrets is therefore $512m$ bits, which is linear in the number m of attributes.

During the setup phase, each node needs to fetch its m secrets and the public parameters, but the key management overhead is small from the perspective of the TTP. The TTP needs indeed only to set up the public parameters and to store them, as well as one key pair (TTP_{pub}/TTP_{priv}), but it does not need to store all the attributes values of each node (contrary to certification authorities e.g.): trapdoors are efficiently generated upon request. The TTP is therefore lightweight, and the key management overhead is negligible given that it takes place offline and thus does not compete with the opportunistic communication.

Furthermore the proposed security solution does not add significant communication overhead: the size of the message header is linear in the number of attributes that it includes, with or without the security solution, but it remains small in comparison with the size of the payload. The security solution only modifies the attribute values through the PEKS function which has a 1024 bits output, therefore the size of the attribute values increases by a factor four at most, while the size of the payload is not significantly modified during the encryption process.

Finally, from a performance point of view, elliptic curve operations used in all the primitives are of the same order of magnitude as classic asymmetric cryptography, but they are still more expensive than symmetric encryption. Indeed, the most costly operation is the pairing computation: one pairing computation per encryption of payload or header, and one pairing per $Test$ evaluation or payload decryption. A pairing computation requires around 11 ms on a pentium III 1 GHz according to the benchmarks established by Lynn based on the PBC library [22]. In comparison, one 1024 bits RSA

decryption takes around 13 ms. Therefore, this cost is acceptable for small texts (according to Moore's law the computation power of mobile devices should exceed that of a pentium III 1 GHz by the end of 2010), like the values of attributes but it is prohibitive when it comes to encrypting large data, like the payload. To circumvent this obstacle, the sender can define a symmetric data encryption key which can further be encrypted with the encryption mechanism proposed in section 4.3.1. We did not mention this option in the description of the scheme for the sake of clarity, but for practical deployment this option should be implemented.

6. Conclusion

In this paper, we introduced a communication design for opportunistic networks which combines epidemic and context-based forwarding. We focused on the analysis of payload confidentiality and user privacy issues in such a protocol and defined the security primitives required to preserve privacy within trusted communities. These primitives require the use of carefully chosen public functions to ensure both privacy and forwarding operations.

Finally, we presented an original solution which is derived from Identity-Based Encryption and Public Encryption with Keyword Search. The use of identity-based encryption in a multiple attribute setting enforces end-to-end payload confidentiality with no end-to-end key management, while the specific use of PEKS allows intermediate nodes to securely discover partial matches between their profile and the message context while preserving user privacy in the trusted communities assumption. The encryption functions depend on an internal random number, hence the output of the functions changes at each execution, even if the input is the same which defeats dictionary attack and traffic analysis. The solution relies on an offline TTP.

This scheme suits opportunistic networks well, because it has a low storage and computation overhead and it relies on an offline TTP which is not required for the correct execution of the protocol during the communication.

Acknowledgement

This work has been supported by the HAGGLE and SOCIALNETS projects, grant agreement number 27918 and 217141, funded respectively by the EC sixth framework program theme FP6-IST-2004-2.3.4 for Situated and Autonomic Communications and the EC seventh framework programme

theme FP7-ICT-2007-8.2 for Pervasive Adaptation. See http://www.haggleproject.org/ and http://www.social-nets.eu/ for further details.

A. Identity-Based Encryption: construction and proofs

In this appendix, we provide some details on the construction of the basic ID-based encryption scheme by Boneh et al. in [10], and then proof of correctness of IBE used in the multiple attribute setting.

A.1. Basic construction

We use the same notations as in section 4.2.1, and we therefore suppose that the parameters q,G_1,G_2,P,e,H_1 as well as TTP_{priv} and $TTP_{pub} = TTP_{priv} \cdot P$ are already defined.

The encryption process of a message M under identity ID denoted by $IBE(H_1(ID), M)$ is a randomized process:

1. choose a random number $r \in \mathbb{Z}_q^*$ (kept secret), and define $U = rP$,
2. compute $V = H_2(e(H_1(ID), TTP_{pub})^r)$, where H_2 is a second cryptographic hash function $H_2 : G_2 \rightarrow \{0,1\}^n$ (n is the bit length of the messages),
3. output $IBE(H_1(ID), M) =< U, M \oplus V >$.

For the decryption process, the destination with identity ID needs to retrieve its private key ID_{priv} from the TTP: $ID_{priv} = TTP_{priv} \cdot H_1(ID)$. Then the destination can compute $H_2(e(ID_{priv}, U))$ which is equal to V thanks to the bilinearity of the pairing e. Therefore

$$IBD(ID_{priv}, < U, M \oplus V >) = M \oplus V \oplus H_2(e(ID_{priv}, U)) = M.$$

A.2. Multiple attribute setting

To guarantee the confidentiality of the payload, we proposed to use IBE in the multiple attribute setting. The encryption and decryption of the payload of a destination with attributes $\mathcal{H}(M) = ||_{j \in L} A_{D,j}$ are:

- $\mathcal{P}(M') = ENCRYPT\_PAYLOAD(M) = IBE(\sum_{j \in L} H_1(A_{D,j}), \mathcal{P}(M))$,
- $\mathcal{P}(M) = DECRYPT\_PAYLOAD(M') = IBD(\sum_{j \in L} A_{priv_{D,j}}, \mathcal{P}(M'))$.

The security of IBE in the multi-user setting is discussed in section 5.

The correctness of the decryption comes from the commutativity and associativity of the group operation of G_1 and the bilinearity of e.

To be more precise, the encryption of the payload of message M is $\mathcal{P}(M') =< U, \mathcal{P}(M) \oplus V >$, where:

- $U = rP$ for some random $r \in \mathbb{Z}_q^*$,
- $V = H_2(e(\sum_{j \in L} H_1(A_{D,j}), TTP_{pub})^r)$.

For the decryption process to be correct, one just need to verify that V is equal to $H_2(\sum_{j \in L} A_{priv_{D,j}}, U))$

$$\begin{aligned} H_2(e(\sum_{j \in L} A_{priv_{D,j}}, U)) &= H_2(e(\sum_{j \in L} TTP_{priv} \cdot H_1(A_{D,j}), rP)) \\ &= H_2(e(TTP_{priv} \cdot \sum_{j \in L} H_1(A_{D,j}), P)^r) \text{ (eq. (1) with } r) \\ &= H_2(e(\sum_{j \in L} H_1(A_{D,j}), TTP_{priv} \cdot P)^r) \text{ (eq. (1) with } TTP_{priv}) \\ &= H_2(e(\sum_{j \in L} H_1(A_{D,j}), TTP_{pub})^r) \\ &= V. \end{aligned}$$

B. Public Encryption with Keyword Search: construction and proofs

In this appendix, we provide some details on the construction of one of the Public Encryption with Keyword Search (PEKS) scheme by Boneh et al. in [11], and then proof of correctness of PEKS used in the multiple TTP setting.

B.1. Construction based on bilinear pairings

Technically the construction of PEKS has some similarities with IBE, and we describe the scheme in the scenario of figure 3. In addition to the parameters described in 4.2.1, node A chooses a random number in $\mathbb{Z}_q^*$ denoted by A_{priv} which is kept secret, and publishes the public key as $A_{pub} = A_{priv}P$.

B can construct the $PEKS$ of a keyword W_1 with the knowledge of the public key of A by choosing a random number $r \in \mathbb{Z}_q^*$ and then computing:

$$PEKS(A_{pub}, W_1) = (rP, H_3(e(H_1(W_1), A_{pub})^r)),$$

where $H_1 : \{0,1\}^* \rightarrow G_1$ and $H_3 : G_2 \rightarrow Z_q^*$ are two cryptographic hash functions.

A constructs the trapdoor corresponding to keyword W_2 by computing

$$Trapdoor(A_{priv}, W_2) = A_{priv} H_1(W_2).$$

Given a $Trapdoor(A_{priv}, W_2)$ and $PEKS(A_{pub}, W_1) = (U, V)$ the $Test$ function consist in checking whether V and $H_2(e(Trapdoor(A_{priv}, W_2), U))$ are equal or not. If true, it means that $W_1 = W_2$, otherwise it means that $W_1 \neq W_2$. In both cases, C does not discover the values of W_1 or W_2.

B.2. Multiple TTP setting

To protect against a malicious TTP, we proposed to use the modified version of PEKS in a multiple TTP setting. In this case, the encryption of the header $\mathcal{H}(M) = ||_{j \in L} A_{D,j}$ by the source is:

$$\begin{aligned} H'(M) &= ||_{j \in L}(E_j, PEKS(\sum_{k=1}^{w} TTP_{k,pub}, A_{D,j})) \\ &= ||_{j \in L}(E_j, (rP, H_3(e(H_1(A_{D,j}), \sum_{k=1}^{w} TTP_{k,pub})^r))), \end{aligned}$$

where r is a random number in $\mathbb{Z}_q^*$.

To verify the correctness of this operation we denote the tuples generated by the PEKS function as:

$$(rP, H_3(e(H_1(A_{D,j}), \sum_{k=1}^{w} TTP_{k,pub})^r)) = (U_{D,j}, V_{D,j}).$$

The $Test$ function consists then in verifying whether $V_{D,j}$ is equal to $H_3(e(\sum_{k=1}^{w} Trapdoor(TTP_{k,priv}, A_{i,j}), U_{D,j}))$ or not.

We have indeed:

$$
\begin{aligned}
V_{D,j} &= H_3(e(H_1(A_{D,j}), \sum_{k=1}^{w} TTP_{k,pub})^r) \\
&= H_3(e(H_1(A_{D,j}), \sum_{k=1}^{w} TTP_{k,priv}P)^r) \\
&= H_3(e(H_1(A_{D,j}), r(\sum_{k=1}^{w} TTP_{k,priv})P)) \text{ (eq. (1) with } r) \\
&= H_3(e((\sum_{k=1}^{w} TTP_{k,priv})H_1(A_{D,j}), rP)) \text{ (eq. (1) with } \textstyle\sum_{k=1}^{w} TTP_{k,priv}) \\
&= H_3(e(\sum_{k=1}^{w} (TTP_{k,priv})H_1(A_{D,j}), U_{D,j})) \\
&= H_3(e(\sum_{k=1}^{w} Trapdoor(TTP_{k,priv}, A_{D,j}), U_{D,j}))
\end{aligned}
$$

Therefore, $V_{D,j} = H_3(e(\sum_{k=1}^{w} Trapdoor(TTP_{k,priv}, A_{i,j}), U_{D,j}))$ if $A_{i,j} = A_{D,j}$ (or in the improbable case of a collision, see section 5), which proves that the $Test$ function is compatible with the sum operation, and therefore the scheme in the multiple TTP setting is consistent.

References

[1] C. Boldrini, M. Conti, J. Jacopini, A. Passarella, Hibop: a history based routing protocol for opportunistic networks, in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2007, pp. 1–12.

[2] P. Hui, J. Crowcroft, E. Yoneki, Bubble rap: social-based forwarding in delay tolerant networks, in: ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), 2008, pp. 241–250.

[3] H. A. Nguyen, S. Giordano, A. Puiatti, Probabilistic routing protocol for intermittently connected mobile ad hoc network (propicman), in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2007, pp. 1–6.

[4] Haggle project, http://www.haggleproject.org/index.php (2006).

[5] A. Balasubramanian, B. N. Levine, A. Venkataramani, DTN Routing as a Resource Allocation Problem, in: ACM SIGCOMM, 2007, pp. 373–384.

[6] J. Burgess, B. Gallagher, D. Jensen, B. N. Levine, Maxprop: Routing for vehicle-based disruption-tolerant networks, in: IEEE International Conference on Computer Communications (INFOCOM), 2006.

[7] A. El Fawal, J.-Y. Le Boudec, K. Salamatian, Self-Limiting Epidemic Forwarding, Tech. rep., EPFL (2006).

[8] A. Lindgren, A. Doria, Probabilistic routing protocol for intermittently connected networks, in: IRTF Internet Draft, draft-irtf-dtnrg-prophet-00.txt, 2008.

[9] T. Spyropoulos, K. Psounis, C. S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, in: ACM SIGCOMM workshop on Delay-tolerant networking (WDTN), 2005, pp. 252–259.

[10] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: CRYPTO, 2001, pp. 213–229.

[11] D. Boneh, G. Crescenzo, R. Ostrovsky, G. Persiano, Public-key encryption with keyword search, in: Eurocrypt, 2004.

[12] L. Lilien, Z. Kamal, V. Bhuse, A. Gupta, Opportunistic networks: The concept and research challenges in privacy and security, in: NSF Intl. Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN 2006), Miami, 2006.

[13] S. Symington, S. Farrell, H. Weiss, P. Lovell, Bundle security protocol specification draft-irtf-dtnrg-bundle-security-12, http://tools.ietf.org/html/draft-irtf-dtnrg-bundle-security-12 (november 2009).

[14] A. Shikfa, M. Önen, R. Molva, Privacy in context-based and epidemic forwarding, in: AOC 2009. 3rd IEEE International WoWMoM Workshop on Autonomic and Opportunistic Communications, June 15, 2009, Kos, Greece, 2009.

[15] L. Buttyán, J.-P. Hubaux, Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks, Tech. Rep. DSC/2001, EPFL (2001).

[16] P. Michiardi, R. Molva, Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, in: IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security, 2002, pp. 107–121.

[17] M. Önen, A. Shikfa, R. Molva, Optimistic fair exchange for secure forwarding, in: MOBIQUITOUS '07: Proceedings of the 2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking&Services (MobiQuitous), 2007, pp. 1–5.

[18] P. Oechslin, Making a faster cryptanalytic time-memory trade-off, in: CRYPTO, 2003, pp. 617–630.

[19] R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM.

[20] D. Charles, K. Jain, K. Lauter, Signatures for network coding, International Journal of Information and Coding Theory (2009) 3–14.

[21] A. Menezes, S. Vanstone, T. Okamoto, Reducing elliptic curve logarithms to logarithms in a finite field, in: STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing, 1991, pp. 80–89.

[22] B. Lynn, The pairing-based cryptography library, http://crypto.stanford.edu/pbc/ (2006).