Privacy in Context-Based and Epidemic Forwarding

Abdullatif Shikfa
EURECOM
France
Abdullatif.Shikfa@eurecom.fr

Melek Önen
EURECOM
France
Melek.Onen@eurecom.fr

Refik Molva
EURECOM
France
Refik.Molva@eurecom.fr

Abstract

Autonomic and opportunistic communications require specific routing algorithms, like replication-based algorithms or context-based forwarding. Privacy is a major concern for protocols which disseminate the context of their destination. In this paper, we focus on the 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 two refinements of identity-based encryption, namely searchable encryption and policy-based encryption. This new solution enables forwarding while preserving privacy by allowing secure partial matches in the header and by enforcing confidentiality of the payload.

1 Introduction

Context-based forwarding (e.g.[3, 10, 13]) 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 addressing. To be more precise, each message is associated with a message context which corresponds to the context of its destination, and nodes make their forwarding decisions by comparing the context of the message with their own context and the context 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). Context-based forwarding is particularly adapted to challenged heterogeneous environments, like opportunistic and autonomic networks [9], where end-to-end connectivity is not guaranteed.

Indeed, in such environments, classical routing mechanisms are 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. [2, 6, 8, 11, 19]). 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.

Privacy of the destination is a crucial issue in such a protocol. The message context is indeed essentially the context of the destination, and the message is forwarded through various intermediate nodes that may not be trusted by the destination. 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 because of obvious 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. These requirements lead us to searchable encryption and ID-based cryptography. Therefore we propose a new solution based on two refinements of ID-based cryptography, namely Public Encryption with Keyword Search (PEKS) [4] and policy-based encryption [1]. The particular use of these functions 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 Trusted Third Party.

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 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 privacy enforcement.

In the next section, we first describe the context-based and epidemic network model. We define the threat model and then focus on the security requirements of the proposed solution. Section 3 analyzes a basic approach based on hashes, whereas section 4 presents our original scheme based on PEKS and policy-based encryption. 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 adapted to opportunistic and autonomic networks. In such environments, the transmission of messages relies on opportunistic strategies like epidemic forwarding. The main challenge in epidemic forwarding is to find a suitable heuristic in order for the nodes to decide either to 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. 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 is defined as a set of attributes $\{A_j\}_{1 \leq j \leq m}$, where each attribute A_j is a couple attribute name E_j, attribute value V_j. The set of attribute names $\{E_j\}_{1 \leq j \leq m}$ is known by all nodes. The value of the attribute A_j at node N_i is denoted by $V_{i,j}$, and the pair $(E_j, V_{i,j})$ is the attribute j of node i denoted by $A_{i,j}$. Finally, the profile $\mathcal{P}(i)$ of node i is the concatenation of all its attributes: $\mathcal{P}(i) = A_{i,1}||...||A_{i,m}$.

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 $H(M)$ and payload $P(M)$, $M = H(M)||P(M)$. The header $H(M)$ holds the profile of N_D known by N_S:

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

where $L \subset [1, m]$ is the subset of the indexes of the values of $\mathcal{P}(D)$ 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 $\mathcal{P}(i)$ with the header $H(M)$ to extract the subset $Q \subset L$ of indexes of values that are shared between $H(M)$ and $\mathcal{P}(i)$, such that $\forall j \in Q, A_{D,j} = A_{i,j}$. After this subset extraction, N_i can compute the probability of meeting the destination as $p_i(M) = |Q|/|L|$, where $|X|$ denotes the cardinal of a set $|X|$. $p_i(M)$ is used as a metric in our heuristic and N_i decides either to drop the message or to store and forward it depending on $p_i(M)$.

A message can have one or several destinations: a node N_i knows that it is a destination of M if $p_i(H(M)) = 1$. In the sequel of the paper, destination thus designs one node or a set of nodes depending on the header.

This protocol takes forwarding decisions based on the destination context, which is potentially sensitive information. This protocol hence needs to be enhanced with security mechanisms both from a privacy and robustness perspective. To this extent, we first define the threat model and then the security requirements.

2.2 Threat model and security assumptions

The most obvious threat in this protocol is privacy leakage: a nodes' profile should not be exposed to other nodes in clear, even if it helps forwarding. Hence, we first assume that malicious nodes, eavesdropping the communication, or even curious intermediate nodes are interested in discovering the full profile of the destination of a message. Privacy should therefore be enforced, but with some limitations inherent to the protocol. Indeed, from the design of the protocol, an intermediate node should be able to detect matches between its profile and the destination of the message, thus full privacy cannot be assured.

Therefore, we make the assumption of trusted communities, which was first introduced in [18] as intra-community privacy. Communities are defined on attributes' basis: all nodes sharing a given attribute A_j form the community of attribute A_j. In the trusted communities assumption, nodes which belong to the same community trust each other and they do not attack 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.

Furthermore, we assume that nodes might adopt a malicious behavior for a specific message only based on the knowledge they get from the said message. 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. Yet, 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 cooperation enforcement schemes ([7, 12, 15]).

Finally, we assume the existence of a Trusted Third Party (TTP) during the setup phase to distribute security materials (like keys or certificates). Yet, such an authority might not be reachable during the runtime phase because of the delay-tolerant nature of the network during deployment. Hence the TTP is only considered offline.

2.3 Security requirements

As mentioned in section 2.1, the proposed protocol, which combines epidemic forwarding with context-based information, requires that the header of each message includes the context of the destination. This presents a major privacy threat, because the context of the nodes are disseminated throughout the network. A user would agree on sharing all or part of his context information with his friends or some members of his communities but he would certainly be reluctant to see his context accessible to strangers. As pointed out in [16], privacy is expected to be a significant concern for acceptance of pervasive environments, hence this protocol needs to be enhanced with security mechanisms to preserve user's privacy. As mentioned in section 2.2, this requirement is not only critical from a privacy perspective, but also from a robustness point of view. The disclosure of the destination context can indeed lead to a specific DoS attack against a specific attribute or set of attributes.

To be more precise, several security requirements are needed to secure the protocol. First, the node N_S which is sending a message M should encrypt the header of the message $H(M)$ in order to prevent other nodes from accessing the destination's profile. This encryption function should be public, because any node should be able to send a message, even before meeting the destination (or set of destinations). This encryption function should not prevent forwarding though and an intermediate node N_i should still be able to compute the probability $p_i(H(M))$.

Hence, as counterpart to the header encryption mechanism, N_i should have a method to determine whether an encrypted attribute matches its own profile or not. Contrary to the header encryption function, this method should be private, in that only nodes which share an attribute with the destination should be able to detect a match and hence discover the attribute. Nodes, which do not share the attribute, should only learn that they do not share the attribute but they should not be able to determine what the value of the attribute is.

Furthermore, N_S should also encrypt the payload of the message $P(M)$, such that only the destination can decrypt it. Indeed, if intermediate nodes had access to the payload they would infer valuable information about the context of the destination. The difference between the header and the payload encryption is that header encryption should allow intermediate nodes to find partial matches between their profile and the encrypted header, whereas the payload encryption scheme is an all or nothing scheme, where the destination can decrypt the payload and other nodes cannot extract any information from the encrypted payload.

We point out that the term ”encrypt” used to denote a secure encoding of the header 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 the security requirements, the proposed protocol needs the four following security primitives:

1. **ENCRYPT_HEADER**: used by the source to encrypt the header of the message. This function should be public and enable partial matches by authorized nodes.
2. **ENCRYPT_PAYLOAD**: public function used by the source to encrypt the payload of the message for the destination.
3. **MATCH_HEADER**: used by an intermediate node to determine matches between its own profile and the encrypted header of the received message. The intermediate node discovers matching attributes but does not learn the value of non-matching attributes with this private function.
4. **DECRYPT_PAYLOAD**: private function used by the destination to decrypt the encrypted payload.

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 [13]. 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 $H(M)$, thus obtaining:

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

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.

Thanks to the preimage resistance of $hash$, this 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 is seducing 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 2.3 though. This solution is namely 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 simply compute the hash of each word in a dictionary and then identify the values in the message header, to discover the value of the attributes that did not match.

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.

Since ENCRYPT_HEADER has to be implemented by a public function because of the design of the protocol, two important properties are required to avoid dictionary attacks:

- 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 don't change.
- MATCH_HEADER, the counterpart of ENCRYPT_HEADER, should be private. This was already mentioned in section 2.3, but the basic solution does not verify this property.

4 Our proposed solution

4.1 General idea

Our goal is to protect the privacy of communicating nodes by encrypting the header and the payload while not disrupting the forwarding of messages. This implies that nodes with shared attributes are able to match these attributes in the header (partial match) while only the destination can decrypt the payload (complete match). The basic approach emphasized the importance of thwarting dictionary attacks, hence our protocol requires 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 [14]. Yet our problem is different because we require public encryption functions that can be used by any node without the need of a secret, and we therefore cannot rely on shared keys which are required by symmetric techniques. Regarding asymmetric techniques, classical schemes like RSA [17], which are based on certificates to prove a node's identity, are not adapted either because a node would have to fetch a destination certificate before sending a message to this destination, and this is unpractical in a DTN environment.

The alternative to classical asymmetric techniques is identity-based cryptography, which avoids the need for certificates. Therefore we propose a solution based on refinements of ID-based encryption, which we tailored to our needs to allow any node to compute an encrypted version of the message. The encryption functions depend on an internal random number, hence the output of the functions change at each execution, even if the input is the same. The solution relies on an offline TTP. This means that there is a setup phase during which nodes contact the TTP to retrieve some secrets corresponding to their profile, but afterward, during the runtime phase, the TTP is not needed anymore. In the next section, we present an overview of two cryptographic tools and then we focus on the innovative way in which they are integrated in our scheme to meet the privacy requirements.

4.2 Policy-based cryptography and Public Encryption with Keyword Search (PEKS)

ID-based cryptography is a type of public-key cryptography in which the public key of a node is the node's identity. The main advantage of ID-based cryptography is that a node does not need to know the public key (or a certificate) of a destination to send an encrypted message. Boneh et al. proposed the first practical ID-based encryption scheme called IBE in [5], and since then many developments have been proposed. We present two interesting developments to our protocol, namely policy-based cryptography and PEKS.

Policy-based cryptography proposed by Bagga et al. [1] is an extension of ID-based cryptography where the ID used for encryption is replaced by a logical expression of several basic identities. In the particular case of policy-based encryption, this allows to encrypt a packet with a key corresponding to the conjunction or disjunction of two identities or attributes, which permits an easy management of complex security policies. For our protocol, we just need the property that we can encrypt a message with a key corresponding to the conjunction of several attributes.

Another refinement over ID-based encryption is PEKS, introduced by Boneh et al. in [4]. PEKS allows an intermediate node to look for a match on encrypted data, provided the node has some adapted trapdoor. The node does not learn any information except whether a match occurred or not. We give a functional example with three nodes A, B and C to explain the scheme more precisely. If B wants to send a searchable keyword W to A through C, it encrypts W with a specific function $PEKS$ and sends $M = PEKS(A_{pub}, W)$ to C, where A_{pub} is the public key of A. C cannot infer any information on M at this step. If A wants to enable C to look for the keyword W in the messages that are addressed to her, she has to provide C with a trapdoor $T = Trapdoor(A_{priv}, W)$, where A_{priv} is the private key of A. With both M and T, C proceeds to a test $Test(M, T)$ which returns true if T is the trapdoor corresponding to W and M the encryption of the same W and false otherwise. Hence, if the result is true, C only knows that A and B encrypted the same keyword (but it does not learn the value of the keyword W) and if the result is false it just deduces that it was not the same keyword. We defer the practical constructions of $PEKS$, $Trapdoor$ and $Test$ to [4], and we now focus on the description of our solution, and more specifically on the particular use of PEKS and policy-based encryption.

4.3 Description of the scheme

The functions $PEKS$, $Trapdoor$ and $Test$ presented in the previous section are public functions, but $Trapdoor$ requires the private key of the destination as input which is not practical in the DTN scenario. We therefore propose to modify the scheme defined in [4] by introducing a TTP which computes trapdoors for authorized nodes. Our protocol has thus two phases: a setup phase, where nodes retrieve their secrets (analog to key distribution in other schemes) and a runtime phase where nodes are deployed and communicate between each other.

4.3.1 Setup phase

In the setup phase, all nodes have access to a Public Key Generator which is a Trusted Third Party (TTP). TTP generates m pairs of public/private key pairs (one per attribute) $TTP_{pub,j}/TTP_{priv,j}$, for $1 \leq j \leq m$, and gives the m public keys $TTP_{pub,j}$ to all nodes.

Furthermore each node N_i sends his profile $\mathcal{P}(i)$ to TTP, which verifies the validity of the said profile (the method of verification is out of the scope of this paper) and in turn gives to N_i the trapdoors associated with $\mathcal{P}(i)$. To be more precise, N_i receives the set of couples $(E_j, Trapdoor(TTP_{priv,j}, V_{i,j}))$ for $1 \leq j \leq m$, where $Trapdoor$ is the trapdoor function of the PEKS scheme (see section 4.2). These trapdoors will be used to find matches between a received header and N_i's profile.

TTP also provides N_i with the private keys corresponding to its attributes. This is similar to the private key corresponding to a node's ID that is provided in IBE. We call these keys $A_{priv_{i,j}}$, and the corresponding public key is simply $A_{i,j}$ itself. The private keys are used to decrypt messages at the destination. The policy-based encryption of a message M with the key $A_{i,j}$ is denoted by $\{M\}_{A_{i,j}}$, and the decryption with the associated private key verifies $\{\{M\}_{A_{i,j}}\}_{A_{priv_{i,j}}} = M$.

To sum up, at the end of this setup phase, each node N_i has $3m$ secrets which are:

- public keys of the TTP: $TTP_{pub,j}$,
- trapdoors corresponding to $\mathcal{P}(i)$: $Trapdoor(TTP_{priv,j}, V_{i,j})$,
- private keys corresponding to $\mathcal{P}(i)$: $A_{priv_{i,j}}$,

for $1 \leq j \leq m$.

4.3.2 Runtime phase

During the runtime phase, TTP is offline and nodes cannot access it, they only use the keys they retrieved during the setup phase. We now describe our proposal for the security primitives presented in section 2.3:

- **ENCRYPT_HEADER:**

  Consider a node N_S which wants to send a message M to N_D. As described in section 2.1, N_S should include a specific header with the attribute of the destination. In order to protect privacy, N_S sends a searchable encryption of this header by using the public keys $TTP_{pub,j}$ and the $PEKS$ function described in section 4.2. Hence, each attribute $A_{D,j}$ is modified in a searchable encrypted attribute $A'_{D,j} = (E_j, PEKS(TTP_{pub,j}, V_{D,j}))$. N_S is able to compute these $A'_{D,j}$ because the $PEKS$ function is public, and N_S retrieved $TTP_{pub,j}$ during the setup phase. The modified header $H'(M)$ that is actually included by N_S is:
  $$H'(M) = ||_{j \in L} A'_{D,j}$$
  with $L \subset [1, m]$.

- **ENCRYPT_PAYLOAD:**

  In order to complete the protection of privacy, the rest of the message M, which is the payload $P(M)$, should also be encrypted such that only the destination (or the set of destinations) can decrypt it. At this step, we use policy-based encryption (see section 4.2) to encrypt the payload with the conjunction of the attributes of the profile of the destination. To be more precise the encrypted payload $P'(M)$ is

$$P'(M) = \{M\}_{\wedge_{j \in L} A_{D,j}}$$

where $\wedge$ designates the logical and, and $\wedge_{j \in L} A_{D,j}$ is the conjunction of all attributes whose index lies in L.

Finally the message actually sent by N_S to its neighbors is $M' = H'(M) || P'(M)$.

- **MATCH_HEADER:**

  When an intermediate node N_i receives a message M', N_i needs to compute $p_i(M')$ to take a forwarding decision. To this end, N_i has to determine the matches between $H'(M)$ and its own profile. At this step, N_i uses the $Test$ function of PEKS together with the trapdoors it retrieved during the setup phase. To be more precise, for each $j \in L$, N_i extracts $PEKS(TTP_{pub,j}, V_{D,j})$ from $H'(M)$ and computes

  $$\begin{aligned} Test(\quad & PEKS(TTP_{pub,j}, V_{D,j}), \\ & Trapdoor(TTP_{priv,j}, V_{i,j}) \quad), \end{aligned}$$

  which outputs:

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

  When the output is true, N_i knows that it shares the attribute $A_{i,j}$ with the destination and it adds j to the set Q of matching indexes (see section 2.1). On the contrary, if the output is false, N_i does not learn any additional information about $A_{D,j}$. N_i can thus determine the set Q and compute the probability $p_i(M')$ to take a forwarding decision.

- **DECRYPT_PAYLOAD:**

  If $p_i(M') = 1$, it means that N_i is actually a destination of M', hence it needs to decrypt the payload. To this end, N_i needs to construct the decryption key corresponding to $\wedge_{j \in L} A_{D,j}$. This is possible only with the knowledge of $A_{priv_{D,j}}$ for all $j \in L$. Since $p_i(M') = 1$, it means that for all $j \in L$, $A_{D,j} = A_{i,j}$, hence N_i indeed knows $A_{priv_{D,j}}$ for all $j \in L$. N_i can therefore construct the decryption key that we denote by $\wedge_{j \in L} A_{priv_{D,j}}$, and N_i can access the payload of the message by computing:

  $$P(M) = \{P'(M)\}_{\wedge_{j \in L} A_{priv_{D,j}}}.$$

This completes the description of our scheme, and we evaluate its security and efficiency in the next section.

5 Evaluation

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

The security of the encryption schemes we use were formally proved by their authors in [1] and [4], and the proofs essentially mean that these schemes are secure under the assumption that the Bilinear Diffie-Hellman problem is intractable, which is a widely accepted assumption.

Hence, the MATCH_HEADER primitive allows an intermediate node N_i to discover matches between the destination profile and its own profile thanks to the $Test$ function of PEKS. N_i cannot discover the non-matching attributes by using the $Test$ function because the function requires a trapdoor as input, and N_i only receives the trapdoor corresponding to its own profile during the setup phase. Therefore, even though $Test$ is a public function, $Test(., Trapdoor(TTP_{priv,j}, V_{i,j})$ is a private function that only nodes with the correct attribute can evaluate. This primitive reveals the matching attributes, hence privacy is not absolute, but this fits under the trusted communities assumption described in section 2.2.

Thanks to the properties of policy-based encryption, the DECRYPT_PAYLOAD primitive requires a private key that is computable only by nodes which have all the attributes of the header. These nodes are, by design of the protocol, the destinations of the message. Other nodes cannot deduce any information about the payload even if they share some of the attributes.

Furthermore, ENCRYPT_HEADER and ENCRYPT_PAYLOAD use only public functions and public keys that are distributed during the setup phase to all nodes. Any node N_S can therefore send messages, even before meeting the destinations during the runtime phase. These primitives also avoid the dictionary attack trap, because they make use of internal randomization, hence their output is different at each execution, even if the inputs do not change.

Hence the proposed framework ensures privacy against curious nodes or eavesdroppers, while enabling the computation of the probability used in forwarding decisions.

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. As in many DTN protocols, key revocation is a difficult problem. In order to reduce the impact of this issue, we propose that, periodically, the TTP regenerates keys and all nodes have to do setup to get updated keys.

Concerning storage, as mentioned in section 4.3.1, each node has to store $3m$ secrets. Each of these secrets is in fact an element of a group of points on an elliptic curve of prime order p. In such settings it is sufficient to have p of 160 bits length to have a security equivalent to 1024 bits RSA. The storage overhead is therefore $480m$ bits, which is linear in the number m of attributes.

Finally, from a performance point of view, elliptic curve operations used in all the primitives are cheaper than classic asymmetric cryptography but they are still more expensive than symmetric encryption. The cost is acceptable for small texts, 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 use a symmetric encryption algorithm to encrypt the payload with a secret key, and encrypt the secret key with the policy-based encryption. 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.

To put it in a nutshell, this scheme enforces, at reasonable costs, privacy of the destination profile all the way since intermediate nodes do not know what is the final destination of the information, they just know the shared attributes and hence they can take a forwarding decision.

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 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, while avoiding dictionary attacks.

Finally, we presented an original solution which combines PEKS and policy-based encryption. The specific use of PEKS allows intermediate nodes to discover partial matches between their profile and the destination profile, while policy-based encryption enforces confidentiality of the payload. This scheme suits opportunistic networks well, because it has a low storage and computation overhead and it relies on an offline TTP only.

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.

References

[1] W. Bagga, R. Molva, and S. Crosta. Policy-based encryption schemes from bilinear pairings. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2006.

[2] A. Balasubramanian, B. N. Levine, and A. Venkataramani. DTN Routing as a Resource Allocation Problem. In *ACM SIGCOMM*, 2007.

[3] C. Boldrini, M. Conti, J. Jacopini, and 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.

[4] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *Eurocrypt*, 2004.

[5] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001.

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

[7] L. Buttyán and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical Report DSC/2001, EPFL, 2001.

[8] A. El Fawal, J.-Y. Le Boudec, and K. Salamatian. Self-limiting epidemic forwarding. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2007.

[9] Haggle project, 2006. http://www.haggleproject.org/index.php.

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

[11] A. Lindgren and A. Doria. Probabilistic routing protocol for intermittently connected networks. In *IRTF Internet Draft, draft-irtf-dtnrg-prophet-00.txt*, 2008.

[12] P. Michiardi and 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.

[13] H. A. Nguyen, S. Giordano, and 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.

[14] P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In *CRYPTO*, 2003.

[15] M. Önen, A. Shikfa, and R. Molva. Optimistic fair exchange for secure forwarding. In *International Conference on Mobile and Ubiquitous Systems: Networking & Services, (MobiQuitous)*, 2007.

[16] L. Opyrchal, A. Prakash, and A. Agrawal. Supporting privacy policies in a publish-subscribe substrate for pervasive environments. *JNW*, 2007.

[17] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[18] A. Shikfa, M. Önen, and R. Molva. Privacy in content-based opportunistic networks. In *Workshop on Opportunistic Networking (WON)*, 2009.

[19] T. Spyropoulos, K. Psounis, and 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.