Authenticated Resource Management in Delay-Tolerant Networks using Proxy Signatures

Dominik Schürmann*, Jörg Ott† and Lars Wolf*
*Institute of Operating Systems and Computer Networks (IBR), TU Braunschweig, Germany.
†Department of Communications and Networking (Comnet), Aalto University, Finland.
Email: d.schuermann@tu-braunschweig.de, jo@netlab.tkk.fi, wolf@ibr.cs.tu-bs.de

***Abstract*—In Delay-Tolerant Networks (DTN), individual nodes with much higher rates of sending new bundles than average can degrade the delivery rate of other nodes substantially. They have a much higher impact on the overall network fairness than in traditional networks because of DTN-specific properties, such as decentralized design and the store-and-forward approach.**

Authenticated resource management schemes were proposed to guarantee minimum delivery rates in the presence of nodes with high resource utilization as well as in the presence of malicious nodes performing Denial-of-Service (DoS) attacks. They partition the buffer adaptively based on the source node identifier of incoming bundles, which is cryptographically authenticated by the network.

We extend such approaches by using a cryptographic primitive named proxy signature. Our method allows treating a bundle not only based on its source node. Instead, a combined affiliation of the source node together with the requesting node can be used which allows for better support of important communication patterns such as request-response. Our method can improve the overall fairness and is similar to a reverse charge call in telephone networks, as the requesting node "pays" for the response by allowing it to also use buffer space normally assigned to itself. We evaluate our approach using simulations in different scenarios.

I. Introduction

Delay-Tolerant Networks (DTN) have received increased attention due to their application in scenarios without central infrastructures [1] and have introduced new security issues emerging from high communication delays, as well as from decentralized design [2, 3]. For instance, because not all rural areas are covered by local service providers messages have to be send hop-by-hop in a DTN to enable information exchange between communities. Mobile devices are used to carry messages from source to destination resulting in large communication delays. Thus, these networks have special requirements regarding storage buffers. Since these buffers are often limited, buffer management schemes are proposed to improve the overall delivery rate in DTNs [4, 5, 6]. However, most of the proposed buffer management schemes are not designed to prevent Denial-of-Service (DoS) attacks. DoS attacks on DTNs have a high impact on the overall performance due to the unique properties of such networks [7, 8]. A simple attack on storage buffers by sending many big bundles of various payloads using different forged IDs (Sybil attack) causes a significant performance loss. In absence of an authentication scheme, it will result in full capacity utilization of storage buffers. Thus, a small subset of malicious nodes can have a substantial negative impact on the delivery rate of benign bundles.

This work was done while Dominik Schürmann was a guest at the Department of Communications and Networking, Aalto University, Finland.

A scheme that addresses this problem is proposed by Solis *et al.* [9]. Their "fine-grained buffer management" drops bundles from the buffer based on thresholds that are defined for each source identifier of stored bundles. To identify nodes, they rely on already deployed authentication techniques. The thresholds can also be based on network domains, which constitute specific subsets of the entire network. Their "coarse-grained buffer management" separates messages into ones coming from insiders or outsiders based on the domain affiliation of the source node. The authors propose a system using pre-emptive buffer management to prioritize messages from insiders over messages from outsiders, by setting the thresholds accordingly. They showed significant improvements in dealing with resource hogs using their schemes.

As mentioned in their future work, their scheme lacks fairness when operating under specific communication patterns, e.g., request-response patterns. Fairness means that there should be no way for individual nodes to obtain advantages, in this case higher utilization of buffer space, compared to other nodes, while decreasing their utilization. In this case, decreasing their utilization means that an allocation of buffer space would lead to discarded bundles originating from others. Pre-emptive buffer management by Solis *et al.* fulfills this definition, when considering unidirectional bundles only. Their algorithm drops bundles affiliated to nodes whose buffer space exceeded the most, after an incoming bundle was previously added to the set of bundles and would result in an overfilled buffer. Their scheme lacks fairness when considering the following request-response scenario: All nodes are part of one domain and storage buffers are adaptively partitioned to prevent malicious insiders. Node A sends a request bundle to node B, which gets forwarded hop-by-hop until it reaches its destination B. Because the request bundle is cryptographically authenticated using a signature from A, it will use buffer space intended for A on nodes forwarding it. The response bundle is signed by B and will thus use buffer space intended for B, even though it was requested by A. Node A has an advantage compared to other nodes, as buffer space assigned to B is used for data intended for A. The

scenario is not fair because A achieved higher utilization and caused a decrease of possible utilization by B. It gets worse when malicious nodes are present and overuse the network by sending many requests to B performing a DoS attack. This generates many responses by B intended for malicious nodes. Thus, buffer space needed to answer legitimate requests is blocked on forwarding nodes.

Our work is motivated by Solis *et al.*, but other resource management schemes that utilize variants of Weighted Fair Queuing (WFQ) and buffer management are also able to embrace this scheme. The proposed extension is based on communication patterns that are cryptographically backed by proxy signatures [10]. Using proxy signatures, node A can delegate its signing capability to B, allowing B to also sign the response bundle in the name of both A and B. Thus, we are using proxy signatures to extend the affiliation of a bundle to also provide a reference to the requesting node, which can be verified by any node forwarding the bundle hop-by-hop. Application of such signatures can be found for example in mobile agents [11, 12], grid computing [13], and distributed systems [14]. Proxy signatures have not been used so far to extend the affiliation based on the communication patterns.

In the next section we will give a short introduction into authentication techniques in DTNs, as they provide the basis for any buffer management based on the identification of nodes. In section III, a suiteable proxy signature scheme is proposed. Our own contributions consist of a communication pattern-based application of proxy signatures for DTNs (cf. section IV) and corresponding resource management schemes (cf. section V). Before concluding the work, some applied scenarios are evaluated in section VI.

II. Authentication Techniques

For authenticated resource management, we need signatures that map one-on-one to IDs. The RFC *Bundle Security Protocol Specification* [15] defines methods for encryption and authentication in DTNs. Encryption is implemented using the Payload Confidentiality Block (PCB), while the Payload Integrity Block (PIB) defines digital signatures for authentication. For our purpose, a traditional Public Key Infrastructure (PKI) serves just as well as an infrastructure using Identity Based Cryptography (IBC) [16, 17] as long as it provides authenticated IDs with strong unforgeability [18].

A. Combining Confidentiality with Authenticity

Typical ways to obtain end-to-end confidentiality in combination with authenticity are Sign-then-Encrypt and Encrypt-then-Sign schemes. Using Sign-then-Encrypt results in end-to-end encrypted signatures that need to be decrypted first before verification. Signatures can only be verified by the destination node, as this is the only node that can decrypt the bundle. Thus, for buffer management schemes it is essential to use the latter, as every node forwarding a bundle hop-by-hop should be able to verify the signature and check the affiliation of that bundle. Our choice is contrary to the *Bundle Security Protocol Specification*, where Sign-then-Encrypt "[...] is generally RECOMMENDED and minimizes attacks that, in some cases, can lead to recovery of the encryption key" [15].

The PCB can be implemented using PCB-RSA-AES128-PAYLOAD ciphersuite to provide symmetric AES encryption using a session key derived from RSA based asymmetric cryptography [15]. This authenticated encryption scheme provides the guarantee that the message has not been altered on the way from the security-source to the destination. It is based on AES-GCM, which combines encryption and authentication without the use of asymmetric signatures. AES-GCM generates a ciphertext and a corresponding tag and therefore an Encrypt-then-Authenticate scheme. This ordering follows the recommendation of Krawczyk [19], and no node on the way between its source and destination can alter the tag in a manner that leaves it valid. The symmetric key used for AES-GCM is finally encrypted using the public key of the destination. Using this scheme without PIBs does not include a way to identify nodes, as no signatures are attached. Adding the ID of the originating source to the message before encryption would also not prevent malicious nodes from including IDs that are not their own, when generating a message. Consequently, the scheme has to be used in conjunction with a PIB to get verifiable nodes.

However, the naïve inclusion of a PCB into a surrounding PIB to implement Encrypt-then-Sign leads to a widely known problem in cryptology [20]. A node, for example T, located on the way from security-source A to security-destination B, could replace the PIB with its own block. This will make security-destination B think that the message comes from T and not A.

We propose implementing a secure scheme using cross-referencing [20]: When utilizing PCB-RSA-AES128-PAYLOAD enclosed by PIB-RSA-SHA256 [15], the PCB should also contain the Endpoint Identifier (EID) of the security-source along with the message. Besides verifying the signature, the security-destination must also determine if the EID matches the owner of the public key used to verify the surrounding signature. If the signature is replaced by a malicious node, the security-destination will reject the message because of the non-matching encrypted EID and corresponding public key used for signature verification.

B. Encrypt-then-Sign with Domain Certificates

In our scenario, a bundle is sent from node A to B, who are members of domain $\mathcal{X}$. Every node has a public/private key pair $\langle pk_i, sk_i \rangle$ and an ID_i, where i indicates the node. A bundle is sent over the network containing a ciphertext c of a message m concatenated with the node's ID for cross-referencing (cf. section II-A) and the corresponding signature σ. In this example, the message m is encrypted with the public key of B, resulting in $c = Enc_{pk_B}(m \parallel ID_A)$, and a signature is generated by $\sigma = Sign_{sk_A}(c)$. A domain certificate $cert_{A \in \mathcal{X}}$ is signed by the trusted authority of domain $\mathcal{X}$.

$$cert_{A \in \mathcal{X}} = \langle pk_A \parallel ID_A, Sign_{sk_{\mathcal{X}}}(pk_A \parallel ID_A) \rangle$$

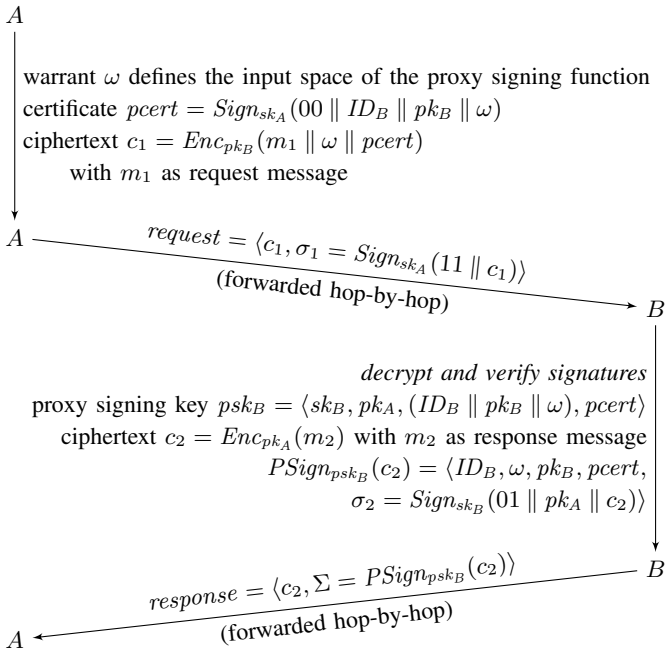


Fig. 1. Delegation-by-Certificate proxy signature scheme

This certificate defines the affiliation with a specific domain and can be verified by using the operation $Verify_{pk_{\mathcal{X}}}(cert_{A\in\mathcal{X}})$. Finally, the resulting bundle will be sent to B.

$$bundle = \langle c, \sigma, cert_{A\in\mathcal{X}}\rangle$$

Every node that forwards this bundle hop-by-hop can verify its affiliation to a source node by $Verify_{pk_A}(c, \sigma)$ and by using the public key pk_A, contained in $cert_{A\in\mathcal{X}}$. The certificate can be verified by using $Verify_{pk_{\mathcal{X}}}(cert_{A\in\mathcal{X}})$, while $pk_{\mathcal{X}}$ has to be in the node's long-time storage of public keys. Consequently, it is verified that the bundle originates from A, which then again is a member of domain $\mathcal{X}$. These public keys could be deployed while bootstrapping the devices or distributed by more sophisticated methods [21]. As discussed in section II-A, finally, B executes $Decrypt_{sk_B}(c \,\|\, ID_A)$ and verifies the cross-reference of ID_A to the signature.

To maintain clarity in the following discussion, we omit domain certificates and the inclusion of IDs used for cross-referencing.

III. Proxy Signatures

Recalling the request-response scenario, our goal is as follows: After A sends a request bundle to B, the response bundle should also be affiliated with A and thus use buffer space intended for requesting node A. Our solution is based on the idea that security-source A could send a proxy signing capability to destination B, which should allow B to also sign the response on behalf of A. The resulting response bundle would therefore be affiliated with both nodes and could use storage space intended only for A, or in more liberal settings, storage space intended for A or B.

We require the typical properties of such proxy signatures, namely *strong unforgeability*, *verifiability*, *strong identifiability*, and *strong undeniability*, as defined by Lee *et al.* [18]. One important requirement is that anyone should be able to distinguish between a proxy signature and a conventional one to verify the sender of the bundle. A standard model for proxy signatures can be securely derived from already deployed digital signatures schemes, for instance RSA-based signatures. The following construction follows the *Delegation-by-Certificate Proxy Signature scheme*, discussed by Boldyreva *et al.* [10]. Other optimized schemes have been proposed [22] and should be chosen based on already deployed cryptographic schemes.

A. Delegation-by-Certificate Proxy Signature Scheme

Following Figure 1, a warrant ω is generated by defining the allowed input space for the proxy signing function. The certificate $pcert$ defines the actual proxy signing capability, consisting of a traditional signature by the delegating node including its ID and public key pk together with ω. The prefixes are added to distinguish between signatures employed in these proxy certificates (using prefix 00), proxy signatures (using prefix 01), and traditional signatures (using prefix 11). This is needed to prevent chosen-message attacks [10], where one node requests a signature of another node for a crafted message, which can be used to forge a proxy signature. Following the scheme, besides the actual request message, node A now also encrypts ω and $pcert$ and sends the signed ciphertext as a request bundle over the network.

After verifying the signature and decrypting the content, node B can construct a proxy signing key $psk_B = \langle sk_B, pk_A, (ID_B \| pk_B \| \omega), pcert\rangle$. The proxy signing mechanism outputs a 5-tuple by

$$PSign_{psk_B}(c_2) = \langle ID_B, \omega, pk_B, pcert, \sigma_2 = Sign_{sk_B}(01 \,\|\, pk_A \,\|\, c_2)\rangle.$$

The response bundle includes the proxy signature $\Sigma = PSign_{psk_B}(c_2)$ that signs the ciphertext and shows an affiliation of the bundle to nodes A and B.

Nodes on the path, forwarding the bundle hop-by-hop, can verify the signature and thus the affiliation of that bundle, using the following checks based on the traditional verification algorithm $Verify_{pk_i}(\cdot,\cdot)$, where pk_i is a public key of node i.

$$\begin{aligned} &PVerify_{pk_A,pk_B}(c_2, \Sigma) = \\ &\quad Verify_{pk_A}(00 \,\|\, ID_B \,\|\, pk_B \,\|\, \omega, pcert) \\ &\quad \wedge\ Verify_{pk_B}(01 \,\|\, pk_A \,\|\, c_2, \sigma_2) \wedge (c_2 \in \omega). \end{aligned}$$

Concluding the example, a verification of the signature Σ would show that node B signed the bundle with a proxy signature from A validly. This means that the bundle is affiliated with both nodes, where it is indisputable that B was the sending node issuing a proxy signature on behalf of A.

IV. Application of Proxy Signatures

For our application of proxy signatures, we define different ways of restricting the usage/validity of the proxy signature.

These are added by defining additional restrictions to be included in the proxy certificate $pcert$ and verified later on.

Validity restriction: This restriction consists of a timestamp, defining the time the proxy certificate was issued and its maximum lifetime. Nodes on the path from source to destination have to verify that the certificate is valid, considering the maximum lifetime in addition to verifying the proxy signature with $PVerify_{pk_i,pk_j}(\cdot,\cdot)$.

Limited responses: Besides including and checking a timestamp, the maximum number of allowed responses can be defined. This requires that all nodes on the path forwarding the bundle hop-by-hop have to keep track of the proxy certificates using a counter of forwarded responses. The certificate would be invalidated if the counter reached a specific limit, resulting in an addition to a certificate blacklist. Expired certificates can be removed from the blacklist using a heartbeat mechanism.

We will concentrate on the utilization of the validity restriction mechanism, as it requires no additional storage on nodes forwarding bundles.

The warrant ω can further be defined to restrict the proxy signing capability to specific usages. Considering that the input to $PSign_{psk_B}(c_2)$, namely c_2 also contains an unencrypted identifier of the destination that bundles are send to. For example, if the network consists of nodes $A, B, C,$ and D, c_2 could be defined as $c_2 = ID_A \parallel Enc_{pk_A}(m_2)$ to set bundles destination to A. We could then restrict the usage of the proxy signing capability by defining the warrant as

$$\omega = \{ID_A \parallel x, ID_C \parallel x\}, \text{ whereas } x \in \{0,1\}^*,$$

which makes the proxy signature only valid for the destinations A and C. Applying this, a delegating node A can restrict the usage of the proxy signature for the proxy node B so that it can only send the response bundle to A or C while leaving the signature valid.

A. Application of Proxy Signatures

Signature delegation can be applied in various resources, e.g., buffer management schemes to enhance the prioritization of messages, when using specific communication patterns.

When operating in multiple domains, domain certificates need to be included in bundles to show the affiliation of one bundle to their domain, as discussed in section II-B. In addition to the inclusion of $cert_{B\in\mathcal{Y}}$, when B sends a bundle signed with a proxy signature delegated by A, it also needs to attach the received $cert_{A\in\mathcal{X}}$. This is needed to show the affiliation of both cooperating nodes to every other node on the path, forwarding this bundle.

The new methods provided for using proxy signatures are defined as follows:

Delegating signing capability: This method performs the steps of generating a warrant ω and a proxy certificate $pcert$ from node A to B (cf. Figure 1).

Generating a proxy signature: Signing a ciphertext c using the delegated signing capability, consisting of $pcert$ and ω, is done by invoking the function $PSign_{psk_B}(c)$, whereas psk_B is the proxy signing key (cf. Figure 1). In addition to node B, the proxy certificate $pcert$ states that it also belongs to node A.

Verification of proxy signature: Verification of proxy signatures is done by $PVerify_{pk_A,pk_B}(c,\Sigma)$, where pk_A is the public key of the delegating node and pk_B is the public key of the node that does the proxy signing. The domain affiliation of the proxy signer B and the delegating node A is verified via the domain certificates also attached to the bundle.

To counter the chosen-message attack, the traditional signing and verification algorithms are extended through the following techniques:

Generating a signature: The traditional signing method is changed by prepending 11 to the ciphertext c, resulting in $Sign_{sk_B}(11 \parallel c)$.

Verification of a signature: Verification of traditional signatures is done by $Verify_{pk_B}(11 \parallel c, \sigma)$.

B. Interoperability in Heterogeneous Networks

To provide interoperability between our new methods and the established standard [15], we need to make some adjustments.

1) It is crucial to make a distinction between traditional signatures, proxy signatures, and proxy certificates by using the prefixes 00, 01, and 11. As this would break signatures generated by legacy nodes, we propose using different hashing algorithms to differentiate between proxy signatures/certificates and traditional signatures. For example, we could use SHA-512 instead of SHA-256, as originally defined by the ciphersuite PIB-RSA-SHA256 [15]. When generating the proxy certificate or proxy signature, the new ciphersuite with SHA-512 is used in conjunction with the prefixes 00 or 01. Conversely, traditional signatures can keep using the existing ciphersuite with SHA-256 without any prefix. This would also provide a strict distinction between signatures, because a node can only request signatures using the ciphersuite with SHA-256, when performing a chosen-message attack. Legacy nodes can now ignore signatures with the new ciphersuite and PIB-RSA-SHA256 will work as before, making the scheme interoperable.
2) When a node without the new algorithms encounters a proxy signature Σ and cannot verify it, there is no signature at all. For the case when new algorithms are deployed and legacy nodes are present, every bundle should also include a traditional signature σ besides Σ.

V. Resource Management using Proxy Signatures

Most resource management schemes abstract from IDs and concentrate on queue management and message scheduling. They provide efficient drop and scheduling policies and are not designed to prevent attacks on the ID of nodes, e.g., pretending to be another node or forging many new IDs (sybil attack).

Dropping policies, for example, Drop-Least-Recently-Received (DLR), Drop-Oldest (DOA), Drop-Front (DF), and schemes like MOFO, MOPR [23] do not utilize IDs and are, when deployed alone, vulnerable to DoS attacks. History-Based

Drop (HBD) [4] uses IDs to maintain a history of encountered nodes, carrying different messages with a specific state. Without authentication, the history is vulnerable to flooding attacks by forging many new IDs (sybil attack).

Only few resource management schemes have been proposed to improve the overall fairness in the presence of a few nodes overusing the network maliciously. Besides reputation and credit-based schemes [21], Solis *et al.* [9] have proposed a pre-emptive buffer management algorithm, taking malicious insider nodes into account. When storage space exceeds its maximum on new incoming bundles, old bundles are dropped from the buffer by defining thresholds for each source identifier. This paper uses buffer management schemes as an example, but other mechanisms could also be constructed in a similar fashion.

Considering communication patterns backed by cryptographic proxy signatures, we can improve the schemes and provide the necessary tools to provide better fairness while preventing DoS attacks.

A. Communication Patterns

By using proxy signatures, various communication patterns are now supported that were not possible using traditional signatures.

One-time request-response: This pattern consists of a request bundle and multiple, potentially fragmented, response bundles. It basically uses one proxy certificate *pcert* that is valid for a lifetime in which the requesting node expects response bundles. The warrant ω can be defined to restrict the destination of the response bundles to the requesting node or to a number of nodes defined in the request bundle that should receive the responses.

Publish-subscribe: Node A could subscribe to information another node B generates, ranging from standardized RSS feed to continuous sensor data. We propose issuing long-term proxy certificates to B. These have to be pro-actively renewed in good time by A to extend the subscription for a desired time frame.

Two-way communication: Two nodes, A and B communicate with each other. They both should issue a proxy certificate, valid for the expected duration of the conversation, to the other node. The proxy certificates should also be pro actively renewed to extend the communication.

B. Integration

We propose an application of the algorithms (cf. section IV-A) in conjunction with "fine-grained buffer management" [9].

1) Request-Response Scenario: A node T that is located at a gateway position (cf. Figure 2) will have to manage many incoming bundles. Consider a request-response communication pattern with one server node S that receives many requests by the nodes R_1, R_2, and R_3. Without proxy signatures, buffer space on node T, intended for S, will reach its maximum utilization fast, because many response bundles are sent by this node. Proxy signatures would vastly improve the situation, as

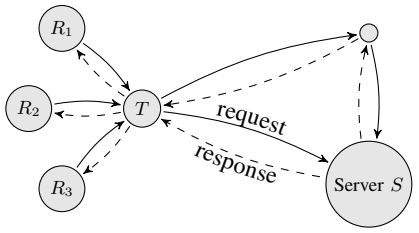


Fig. 2. Request-response scenario with requesting nodes R_1, R_2, R_3, a server S, and a node T in between, storing and forwarding incoming bundles

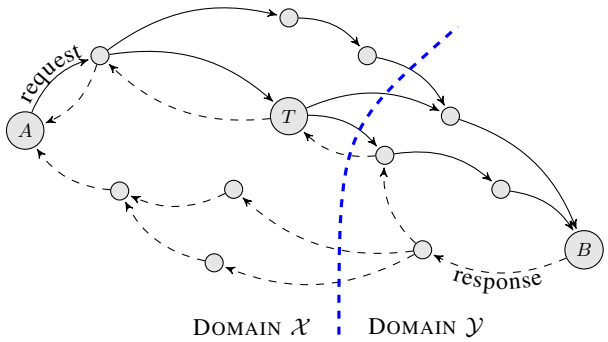


Fig. 3. Request-response scenario for communication between nodes in different domains. $A \in \mathcal{X}$ and $B \in \mathcal{Y}$ with a node $T \in \mathcal{X}$ at the border of the domains

the requesting nodes can delegate their proxy signing capability to S, which results in response bundles assigned to these requesting nodes.

2) Request-Response Scenario with Domains: Considering an example with two domains and nodes A and B communicating between domains (cf. Figure 3), our idea would provide the desired affiliation of response bundles. Node A would issue a proxy certificate to B, providing it with the capability to respond in its name. Bs response, crossing domains from $\mathcal{Y}$ to $\mathcal{X}$, can now be recognized as an intended response affiliated with A and thus get higher priorization on nodes in $\mathcal{X}$ forwarding the bundle hop-by-hop. It is important to point out that the response bundle is also still affiliated with domain $\mathcal{Y}$ and is therefore forwarded normally by nodes in $\mathcal{Y}$. Real-world domain scenarios will differ from the depicted one, as nodes from different domains are not always geographically separated and can also be mobile devices, resulting in highly diverse scenarios. This scenario can also be extended straightforwardly to support more than two domains.

C. Attack Scenarios

Below, we provide an overview over possible attack scenarios, connected to misuse of proxy signatures: Node $A \in \mathcal{X}$ delegates its signing capability to $B \in \mathcal{Y}$ and $C \in \mathcal{Y}$ using the proposed scheme.

Misuse of signing capability: B could misuse the signature delegation and try to flood the storage buffers of nodes in domain $\mathcal{X}$ by sending many messages in the name of A.

Naïve competing nodes: When both nodes B and C are responding in the name of A using proxy signatures at the

same time, they eventually compete for storage space that is kept free for A by other nodes in $\mathcal{X}$.

Malicious competing nodes: In addition to the scenario above, node B could behave maliciously and send many more bundles in the name of A, than other nodes, for instance C. This would decrease the delivery rate of bundles from other nodes that are sending responses to A.

As a first approximation, we could deploy a more fine-grained buffer management scheme: The buffer on nodes forwarding these bundles should be adaptively partitioned by the requesting nodes (in this case A) first. Each of these storage spaces is now again adaptively partitioned by the responding node (B and C). Response bundles are recognized by the attached proxy signatures instead of traditional ones. This finer partitioning scheme could lead to a minimum ensured delivery rate, even if nodes perform Denial-of-Service attacks. As this finer scheme introduces more complexity, we propose its usage only in networks in need of high DoS prevention, such as networks connecting critical infrastructure. We argue that in most networks, A trusts the responding nodes B and C sufficiently, because it already defined them as bundle destinations, whereas A does not trust nodes forwarding this bundle hop-by-hop in most cases. Thus, the attack scenarios and the proposed finer-grained scheme are only needed in networks where the requesting nodes can not trust the responding ones.

VI. Evaluation

We simulated our approach with the ONE simulator [24]. Our implementation[1] consists of a threshold-based dropping scheme, where thresholds are assigned to each source identifier of bundles in the current storage. The thresholds are adaptively balanced to sum up to 1, as we consider the scenario where all nodes are part of one domain (cf. section V-B1). On new incoming bundles, dropping candidates are chosen based on which ones most exceed the threshold. From these dropping candidates, the oldest one is dropped by the simulator, and the procedure is repeated until there is enough space to store the incoming bundle.

A. Simulation Parameters

We have done our simulation based on Helsinki city's central area. The Shortest Path movement model was used to simulate 200 nodes interacting while moving at 0.5 to 1.5 m/s. When sighted within 10 m of reach, bundles are transmitted at 250 kB/s. This approximates a Bluetooth connection and results in short contact times. When nodes reach their destination, they stop for 250 s before moving along.

We chose Spray-and-Wait as the routing model with the default parameters in ONE. Thus, it is configured in binary mode and starts with six copies of each bundle.

Every simulation is repeated 5 times with different seeds for the pseudo random number generator and simulates 12 hours. The standard deviations are depicted in the figures.

[1]Source code of our application class is available at https://github.com/dschuermann/the-one-resource-management

B. Scenario Parameters

The implementation was used to verify the request-response scenario considered in section V-B1, where all nodes reside in one domain and send requests to servers that form a small subnet. To define this subnet, 5 % of the nodes are randomly chosen as servers and tagged accordingly. Three types of bundles exist in the network: requests, responses, and unidirectional bundles. Requests are always sent to one of the servers, which answer with response bundles. Unidirectional bundles are sent to another node of the network without awaiting a response.

Request bundles have a size of 1 kB to 100 kB, response bundles are between 100 kB and 1 MB, and unidirectional bundles are between 1 kB to 1 MB, where all values are chosen uniformly at random. The Time To Live (TTL) is set to 5 hours before bundles are dropped by the ONE simulator. Storage buffers on nodes are chosen to be 5 MB, while servers are considered as devices with more storage and thus have 50 MB storage space. Proxy signatures are simulated by introducing a different categorization of incoming bundles that have to be forwarded and thus stored in buffers. Instead of increasing the threshold for the source identifier of that bundle in the buffer management, the threshold of the destination identifier is increased.

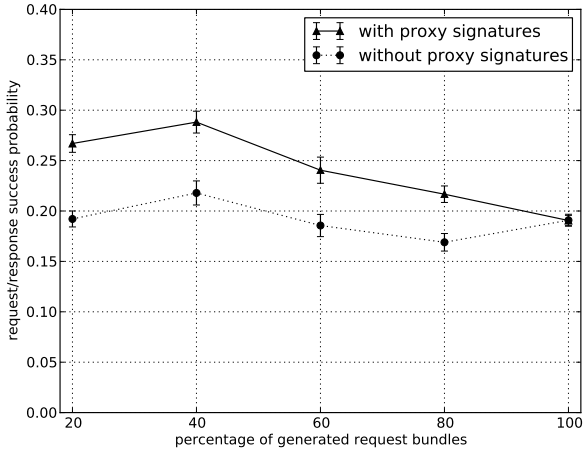
C. Request-Response Success Probability vs Percentage of Generated Requests

We have conducted four main simulations to analyze the request-response success probability depending on the percentage of generated requests. We distinguish between benign nodes that generate a new outgoing bundle every 1000 s simulation time, and malicious nodes with a generation interval of 10 s. Additionally, the probability that a node generates either a new request that is directed to a server or a new unidirectional bundle is parameterized differently in the four studies. When a server receives a request, it generates a new response bundle immediately.

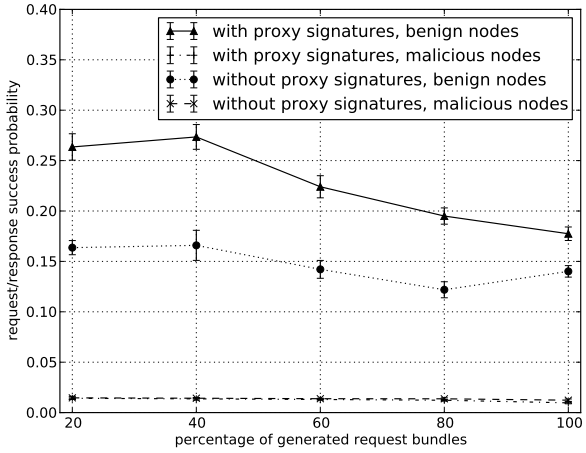
1) Only Benign Nodes: The first two simulations consist of benign nodes only. The probability of a node to generate a new request or a unidirectional message was increased from 20 % requests (80 % unidirectional bundles) to 100 % (only requests, no unidirectional bundles). One simulation was conducted using the pre-emptive buffer management algorithm [9] without the use of simulated proxy signatures, and another with simulated proxy signatures. The resulting request/response success probability is presented in Figure 4a.

Without simulated proxy signatures, incoming requests and unidirectional bundles on forwarding nodes often lead to dropped response bundles because response bundles are often affiliated with candidates in buffers with highly excessive thresholds. These candidates are mostly server identifiers due to the amount of response bundles forwarded hop-by-hop, which are all affiliated with a few servers.

When affiliating response bundles to the destination node by enabling simulated proxy signatures, fewer response bundles are dropped. On nodes forwarding a bundle, incoming response



(a) Only benign nodes



(b) 95 % benign and 5 % malicious nodes

Fig. 4. Request-response success probability depending on the percentage of request bundles. The simulations were conducted using the pre-emptive buffer management algorithm by Solis *et al.* [9] without proxy signatures as well as with simulated proxy signatures.

bundles are now assigned to the destination node and not to one server node. This results in a higher request/response success probability. Response bundles are now distributed over the thresholds of the candidates that result in a more balanced dropping behavior, as requests and unidirectional bundles assigned to a candidate are now often dropped to make space for incoming response bundles. Nevertheless, under high load, there is a competition for buffer space, as the threshold of a candidate is often already high due to several other bundles. When only request bundles are generated (100 % case), proxy signatures do not make a significant difference, as they do not have to compete with unidirectional bundles for buffer space.

2) Malicious Nodes: These simulations consist of 95 % benign and 5 % malicious nodes, who compete for buffer space. Simulating the impact of smaller request intervals, malicious nodes generate new requests every 10 s simulation time instead of 1000 s. One setup was simulated with proxy signatures and one without, as presented in Figure 4b. Comparing Figure 4a with Figure 4b shows the impact of malicious nodes on the request/response success probability when implemented without proxy signatures. Without simulated proxy signatures, malicious nodes have a higher impact on the dropping rate of response bundles by benign nodes. As malicious nodes send more requests, they also cause more response bundles assigned to them that are generated by server nodes. As these response bundles are affiliated with the server, other response bundles, probably for benign nodes, are more often dropped on forwarding nodes. The success probability for malicious nodes are low in both simulations, because of their high request generation interval of 10 s.

Changing the affiliation of a response bundle to the destination node results in better balanced thresholds and fewer responses drops for benign nodes. Thus, especially in scenarios with high load due to a small amount of malicious nodes, proxy signatures can provide a method to maintain a minimum amount of request/response success probability.

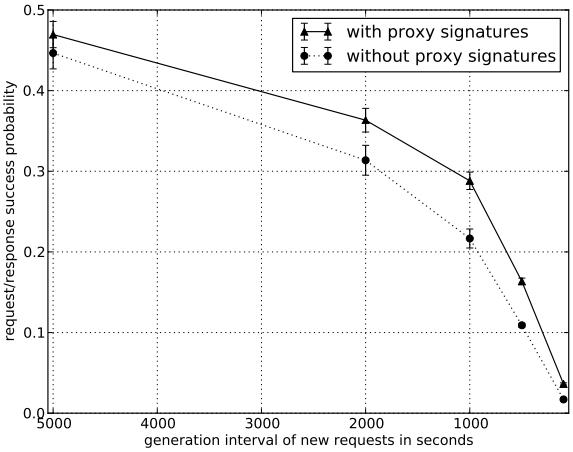


Fig. 5. Impact of generated load on request-response success probability

D. Request-Response Success Probability as a Function of the Offered Load

We conducted two simulations to show the impact of network load due to different generation intervals on the request-response success probability. The basic parameters were set as discussed before to simulate a 12 h work day. These simulations consists of only benign nodes with a fixed 40 % probability to generate requests. This was chosen to simulate a scenario with enough unidirectional bundles that compete with the requests for buffer space. The simulations were conducted with and without simulated proxy signatures. We decreased the request generation interval from 5000 s to 100 s to simulate different network loads.

Figure 5 shows an improvement on the request-response

probability. The highest improvements using proxy signatures result from generation intervals starting with 2000 s. Comparing absolute values from the simulations, the probability that a request is received by a server is approximately the same for simulations with and without proxy signatures. This is compatible with the proxy signatures method that only has effects on response bundles in scenarios where all nodes behave according to the protocol and send the same amount of requests. In simulations with high generation intervals, fewer response bundles have to compete on buffer space, which results in fewer effects by the proxy signatures scheme.

As the scheme was done without malicious nodes, the request generation interval is the same for every node. This shows that our scheme improves the balancing of stored bundles even in scenarios where all nodes behave according the protocol and no malicious nodes are present.

VII. Conclusion

We proposed a way to extend the conventional single affiliation of a bundle to an affiliation based on communication patterns between two nodes backed by proxy signatures. This enhances the method of managing node storage buffers based on trust. Because no Trusted Authority is involved in the process of proxy signature delegation between two nodes, our scheme fits into the decentralized structure of DTNs. Using our validity restriction scheme for the proxy certificates, no history has to be kept, as required in other buffer management schemes.

Our scheme could be extended in multiple ways and opens future research avenues. Besides using Delegation-by-Certificate, other proxy signatures were proposed in the literature [22] that probably provide better performance. They need to be evaluated regarding their security and could be selected based on the specific requirements and already deployed schemes. Basically, to provide improved prioritization of bundles in group conversations, multiple proxy signatures, or, for better performance, Aggregate Signature Schemes [10], could be used. To improve the general prevention of DoS attacks, new storage buffer management schemes could be proposed to provide finer priority schemes, based on different trust levels, similar to the Web-of-Trust in OpenPGP. Another issue worth investigating further is the handling of multicast bundles using our scheme.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '03*, Karlsruhe, Germany: ACM, 2003, pp. 27–34.

[2] L. Wood, W. Eddy, and P. Holliday, "A bundle of problems," in *Aerospace conference*, IEEE Computer Society, 2009, pp. 1–17.

[3] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *3rd International Conference on Security and Privacy in Communication Networks, SecureComm 2007*, 2007, pp. 504–513.

[4] A. Krifa, C. Baraka, and T. Spyropoulos, "Optimal buffer management policies for delay tolerant networks," in *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON '08.*, 2008, pp. 260–268.

[5] A. Krifa, C. Barakat, and T. Spyropoulos, "An optimal joint scheduling and drop policy for Delay Tolerant Networks," in *Proc. of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, WOWMOM '08*, Washington, DC, USA: IEEE Computer Society, 2008, pp. 1–6.

[6] Y. Li, M. Qian, D. Jin, L. Su, and L. Zeng, "Adaptive optimal buffer management policies for realistic DTN," in *Global Telecommunications Conference, GLOBECOM 2009*, 2009, pp. 1–5.

[7] M. Y. S. Uddin, K. Ahmed, and H. D. Jung, "Denial in DTNs," Tech. Rep., Jan. 2010.

[8] G. Ansa, E. Johnson, H. Cruickshank, and Z. Sun, "Mitigating Denial of Service Attacks in Delay-and Disruption-Tolerant Networks," in *Personal Satellite Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 43, Springer, 2010, pp. 221–234.

[9] J. Solis, N. Asokan, K. Kostiainen, P. Ginzboorg, and J. Ott, "Controlling resource hogs in mobile delay-tolerant networks," *Computer Communications*, vol. 33, no. 1, pp. 2–10, May 14, 2010.

[10] A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," *Journal of Cryptology*, vol. 25, pp. 57–115, 2012.

[11] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature," in *Proc. of the 6th Australasian Conference on Information Security and Privacy, ACISP '01*, London, UK: Springer, 2001, pp. 474–.

[12] X. Hong, "Efficient threshold proxy signature protocol for mobile agents," *Information Sciences*, vol. 179, no. 24, pp. 4243–4248, 2009.

[13] M. A. Jabri and S. Matsuoka, "Authorization within grid-computing using certificateless identity-based proxy signature," in *Proc. of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10*, Chicago, Illinois: ACM, 2010, pp. 292–295.

[14] J. Leiwo, C. Hänle, P. Homburg, and A. S. Tanenbaum, "Disallowing Unauthorized State Changes Of Distributed Shared Objects," in *In SEC*, Kluwer, 2000, pp. 381–390.

[15] S. Symington, S. Farrell, H. Weiss, and P. Lovell, *Bundle Security Protocol Specification*, RFC 6257 (Experimental), Internet Engineering Task Force, May 2011.

[16] A. Seth and S. Keshav, "Practical security for disconnected nodes," in *Proc. of the First International Conference on Secure Network Protocols, NPSEC'05*, Boston, Massachusetts: IEEE Computer Society, 2005, pp. 31–36.

[17] W. L. Van Besien, "Dynamic, non-interactive key management for the bundle protocol," in *Proc. of the 5th ACM Workshop on Challenged Networks, CHANTS '10*, Chicago, Illinois, USA: ACM, 2010, pp. 75–78.

[18] B. Lee, H. Kim, and K. Kim, "Strong proxy signature and its applications," in *Proc. of the 2001 Symposium on Cryptography and Information Security, SCIS'01*, vol. 2/2, Oiso, Japan, Jan. 2001, pp. 603–608.

[19] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure Is SSL?)," in *Proc. of the 21st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '01*, London, UK, UK: Springer, 2001, pp. 310–331.

[20] D. Davis, "Defective sign & encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML.," in *USENIX Annual Technical Conference, General Track*, USENIX, Sep. 2, 2002, pp. 65–78.

[21] H. Zhu, "Security in delay tolerant networks," PhD thesis, University of Waterloo, 2009.

[22] M. L. Das, A. Saxena, and D. B. Phatak, "Algorithms and approaches of proxy signature: A survey," *I. J. Network Security*, vol. 9, no. 3, pp. 264–284, 2009.

[23] A. Lindgren and K. Phanse, "Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks," in *First International Conference on Communication System Software and Middleware, Comsware 2006*, 2006, pp. 1–10.

[24] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proc. of the 2nd International Conference on Simulation Tools and Techniques, SIMUTools '09*, Rome, Italy: ICST, 2009.