Secure Network Coding in DTNs

László Czap and István Vajda

***Abstract*—The application of network coding can significantly improve the performance of message delivery in delay tolerant networks, assuming all participants behave honestly. However, if some nodes of the network are compromised, the adversary can launch pollution attack and this way can destroy large amount of data with small effort. Current solutions against pollution attack require public key infrastructure, that is often not available in mobile ad-hoc networks. Our proposal allows packets to verify each other, hence an intermediate node can decide whether these packets can be encoded together without authenticating the source.**

***Index Terms*—Network level security and protection, network coding, pollution attack, integrity protection, delay tolerant networks.**

I. Introduction

In delay tolerant networks, network coding is an appealing technique to improve message delivery ratio. Several research results show the benefit of coding in this environment [1]–[5], however all of them operates in benign environment. If we take into account that an adversarial node can launch pollution attack, the performance of the system seriously drops [6], [7]. If a polluted packet is successfully inserted into the network, not only the adversary, but honest participants also disseminate adversarial packets, if they use the polluted packet to produce novel encoded packets. Therefore, polluted packets may destroy the data of honest nodes also.

The problem of pollution attack is widely investigated [8]–[11], but none of the proposed solutions are applicable in DTNs. These solutions require such infrastructural elements as secure channels or public key infrastructure that DTNs usually can not provide. We describe a general observation (see Section III-A) regarding the PKI requirement, and based on this, we propose a simple, yet efficient *weak verification* method to prevent pollution attack in DTNs.

For our proposal we take as a basis the network coding signature scheme of [11], and introduce a method that verifies the network coded packets not independently one by one, but using each other's authentication information. Our method does rely on a PKI. Though our solution does not provide source authentication, it is able to help an intermediate node to decide whether a batch of packets can be encoded together. This way pollution attack is prevented in the network, while data authentication and data dissemination are separated.

II. System Model

We assume an opportunistic mobile ad-hoc network without any trusted entity or public key infrastructure available. Message dissemination is based on opportunistic communication between the nodes of the network. We consider a single source node and one or more destination nodes. The source node transmits a batch of k messages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k \in \mathbb{F}_p^N$ to the destination nodes. The messages share a common identifier id. It is convenient to assume that messages have the same length and they form a properly augmented basis. These messages can be either independent or fragmented parts of a larger message.

Intermediate nodes are allowed to produce and disseminate linear combinations of messages that belong to the same identifier. An encoded packet is a tuple (id, β, M, σ), where β is an encoding vector and

$$M = \sum_{i=1}^{k} \beta_i \mathbf{m}_i.$$

Note that for convenience, we denote separately, but β is the first k elements of M. σ is the authentication information described later. In this letter we do not make any further assumptions on the message encoding and forwarding algorithm. For particular protocols we refer the reader to [1]–[3].

A. Adversary

The adversary can compromise one or more nodes in the network, but not all of them. The source node is assumed to be honest. The adversary has full control over the compromised nodes, she can read the stored keys (if any) and she can behave arbitrarily. We assume that the strength of the adversary is not sufficient to hinder all communication between the source and the destination nodes. Otherwise, it is clearly impossible to provide defense against the attack. Our goal is to provide defense against the byzantine nodes in the networking layer and counteract pollution attack that destroys data of honest nodes. The higher level (in the transport or application layer) authentication of data is out of our scope. At the data dissemination level, the adversary is allowed to be source of data as well.

III. Signatures Without Public Keys

We take as a basis the network coding signature scheme introduced in [11], and we propose a novel method for packet verification. We shall refer to our method as weak verification, because packets verify each other without providing authentication of the source node.

A. Principle of weak verification

The method of weak verification is based on the following observation. To prevent pollution attack, we do not need all functionalities that digital signatures provide. The problem that an intermediate node faces is to decide whether two (or more) packets can be combined together without producing

Manuscript received September 9, 2010. The associate editor coordinating the review of this letter and approving it for publication was I. Maric.

The authors are with the Laboratory of Cryptography and Systems Security, Budapest University of Technology and Economics, 2. Magyar tudósok krt., H-1117 Budapest, Hungary (e-mail: czap@crysys.hu).

Digital Object Identifier 10.1109/LCOMM.2010.01.101682

a polluted packet. Considering the adversary as an additional source, the question is whether the packets originate from the *same* source, and it does not matter *who* the source is.

Digital signatures with public key certificates provide answer to the *who* question also. We show how the signature scheme of [11] can be used without public keys and certificates to relax the authentication of the source while making it possible to decide whether two packets have the same origin. The idea behind is very simple, we check if the two signatures were produced using the same private key. It can be done without accessing the public key by exploiting the property that the decisional Diffie-Hellman problem (DDH) is easy in the group that is used in [11].

B. The $\mathbf{NCS_1}$ scheme

First, we briefly summarize the $\mathbf{NCS_1}$ scheme of [11] that is a basis of our solution. The signature scheme operates over a bilinear group tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \varphi)$, where

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic multiplicative groups of the same prime order p. The discrete logarithm problem is assumed to be computationally unfeasible in these groups,
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable mapping with bilinear and non-degenerating property,
- $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism.

The source has a private key $\alpha \in \mathbb{F}_p$, and a public key pair $(h, u \in \mathbb{G}_2)$, for which $h^\alpha = u$. It uses a homomorphic hash function $\mathcal{H} : \mathbb{F}_p^N \times \mathbb{Z} \to \mathbb{G}_1$:

$$\mathcal{H}(\mathbf{m}_i, id) = \prod_{j=1}^{k} H(id||j)^{m_{i,N-k+j}} \prod_{\ell=1}^{N-k} g_\ell^{m_{i,\ell}}.$$

Here, $g_1, g_2, \ldots, g_{N-k}$ are publicly known[1] random elements from $\mathbb{G}_1$, $H : \mathbb{Z} \times \mathbb{Z} \to \mathbb{G}_1$ is a cryptographic hash function and N is the length of one message. The signature on a message $\mathbf{m}_i \neq \mathbf{0}$ with identifier id is

$$\sigma_i = \mathcal{H}(\mathbf{m}_i, id)^\alpha. \tag{1}$$

Just like the hash, the signature also has homomorphic property. For a signed packet $(id, \beta, M = \sum_{i=1}^{k} \beta_i \mathbf{m}_i, \sigma)$ the signature is

$$\sigma = \prod_{i=1}^{k} \sigma_i^{\beta_i}. \tag{2}$$

The verification of a signed packet is successful, if $e(\sigma, h) = e(\mathcal{H}(M, id), u)$.

C. Weak verification

Beside the above properties we require the existence of a function $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ that is an efficiently computable isomorphism. Note that the requirement of function ψ does not change the properties of the signature, it only restricts the selection of elliptic curves over which this special bilinear group tuple can be defined.

[1]These parameters are not part of the public key, because all sources of the network share the same values. We can assume that all nodes of the network know them.

According to our method, nodes compute signatures using the $\mathbf{NCS_1}$ scheme regularly (based on eq. (1) and (2)), it is only the process of verification that differs.

Assume an intermediate node receives two packets with the same identifier $(id, \beta_1, M_1, \sigma_1)$ and $(id, \beta_2, M_2, \sigma_2)$. It wants to decide whether they come from the same originator and hence whether they can be combined together. The node performs the following check, without the need to access any public keys:

$$e(\mathcal{H}(M_1, id), \psi(\sigma_2)) = e(\sigma_1, \psi(\mathcal{H}(M_2, id))).$$

If this equation holds, the intermediate node can be sure that the two packets come from the same source, hence they can be combined together without causing pollution. We note that the intermediate node can not decide whether the source of the packets is indeed the stated sender, nor that which of the two packets is honest in case the verification fails. This is because the adversary is treated as a source node and we only answer the question whether the sources of the packets are the same.

Note that weak verification does not require any public keys. Key management hence becomes extremely simple: source nodes can pick a new random signing key for each batch without any further management operations to perform.

In the following theorem we prove that the weak verification method provides the stated properties. A packet $(id, \beta^*, M^*, \sigma^*)$ is forged if its weak verification with an honest packet (id, β, M, σ) succeeds, but M^* is non-zero and $M^* \neq \sum_{i=1}^{k} \beta_i^* \mathbf{m}_i$. Clearly, such packet causes pollution if it is combined together with the honest packet.

Theorem 1: An adversary can not produce a forged packet (as defined above), assuming $\mathbf{NCS_1}$ is a secure network coding signature scheme.

Proof: Indirect. We show that if an adversary produces a forged packet $(id, \beta^*, M^*, \sigma^*)$, it also passes the verification of the $\mathbf{NCS_1}$ signature, hence it is a forged packet in the $\mathbf{NCS_1}$ scheme too.

Let $\alpha^* \in \mathbb{F}_p$ be the number for which

$$\mathcal{H}(M^*, id)^{\alpha^*} = \sigma^*.$$

Such α^* exists, because $M^* \neq \mathbf{0}$, thus $\mathcal{H}(M^*, id) \neq 1$. We show that α^* equals α, the signing key of the honest packet (id, β, M, σ). The following holds:

$$\begin{aligned} e(\mathcal{H}(M, id), \psi(\sigma^*)) &= e\left(\mathcal{H}(M, id), \psi(\mathcal{H}(M^*, id)^{\alpha^*})\right) = \\ &= e\left(\mathcal{H}(M, id), \psi(\mathcal{H}(M^*, id))^{\alpha^*}\right) = \\ &= e(\mathcal{H}(M, id), \psi(\mathcal{H}(M^*, id)))^{\alpha^*} \end{aligned}$$

Similarly,

$$e(\sigma, \psi(\mathcal{H}(M^*, id))) = e(\mathcal{H}(M, id), \psi(\mathcal{H}(M^*, id)))^{\alpha}.$$

From the success of the weak verification it follows that the two expressions equal.

The non-degenerating property of e implies that $e(a, b) = 1$ only if either $a = 1$ or $b = 1$. If the result is 1, $\mathcal{H}(M^*, id) = 1$ or $\mathcal{H}(M, id) = 1$, which occurs only if $M^* = \mathbf{0}$ or $M = \mathbf{0}$, that is not allowed. The result of the mapping thus does not equal 1.

Group $\mathbb{G}_T$ is a cyclic group and has prime order, hence $e\left(\mathcal{H}\left(M, id\right), \psi\left(\mathcal{H}\left(M^*, id\right)\right)\right) \neq 1$ is a generator element of $\mathbb{G}_T$. Thus, $\alpha^* = \alpha$ follows.

This reasoning implies that the forged packet passes the verification of $\mathbf{NCS_1}$ while it is not a valid packet. Hence, an adversary that can forge the weak verification can also forge $\mathbf{NCS_1}$. ■

IV. Extensions

A. Batch verification

In the previous section we introduced the weak verification method for two packets. If a node needs to verify a batch of packets with the same id, the task is to separate packets based on their originators. With the following straightforward algorithm packets from a batch B can be separated into the required number of originator sets $O_1, O_2, \ldots, O_r$. Originator sets consist of packets that can be encoded together without causing pollution. The verifier performs the weak verification on pairs and if the verification fails, a new originator set is created. The algorithm is then restarted on the novel set until no new sets are created. The Algorithm 1 lists the pseudo-code of the batch verification algorithm.

Algorithm 1 Batch verification

```
function batch_verification(B, i = 1)
if sizeof(B) ≤ 1 then
    return
end if
let O_i = B
randomly select an element p_1 from O_i
for each elements p_2 of O_i \ {p_1} do
    if weak_verification(p_1, p_2) fails then
        let O_i = O_i \ {p_2}, O_{i+1} = O_{i+1} ∪ {p_2}
    end if
end for
batch_verification(O_{i+1}, i + 1)
return
```

The above algorithm is rather pessimistic. If there is no attack (presumably in most of the cases) it runs the weak verification $s-1$ times on a batch of size s. The verifier can decide whether there are *any* polluted packets in the batch more efficiently in the following way.

The verifier produces two independent random linear combinations from the packets in the batch and computes the corresponding signatures. It then runs the weak verification on the two novel packets. From the properties of the weak verification method, it follows that successful verification implies that there are no polluted packets in the batch. However, if the verification fails, the verifier has no information about which packets belong to the same originator, hence it has to run Algorithm 1 or it can apply some group testing methods to find the originator sets.

B. Other signature schemes

The principle that we described in Section III is general and can be applied to any other (future) network coding signature schemes also. Using the verification algorithm of the signature scheme, the verifier can decide whether the packet was signed with the private pair of the stated (but not certified) public key. Hence, packets that verify with the same public key have the same originator for sure.

In this case, the drawback is that the public key is required for the weak verification and packets need to be verified one by one and then separated into originator sets based on the corresponding public keys. Using the verification process of the signature scheme however has also an advantage, namely that a packet that fails in the verification process can be treated as polluted packet and dropped. Our algorithm does not drop such packets but puts them into separate originator sets.

V. Summary

We introduce a general idea to prevent pollution attacks in DTNs that use network coding without available public key infrastructure. We show that the functionalities that a network coding signature provides without public key certificates are sufficient for the purpose of preventing pollution attack. Based on the signature scheme of [11], we propose the weak verification method that decides whether two packets have the same originator without accessing any public keys.

We also show extensions to the method. The first provides efficient batch verification, while the other shows the generalization of the idea for any network coding signature schemes.

Acknowledgement

The research leading to these results has received funding from the European Community's 7th FP under grant agreement no. ICT-225186 (WSAN4CIP).

References

[1] S. Ahmed and S. Kanhere, "HUBCODE: message forwarding using hub-based network coding in delay tolerant networks," in *Proc. 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2009, pp. 288–296.

[2] Y. Lin, B. Liang, and B. Li, "Performance modeling of network coding in epidemic routing," in *Proc. 1st International MobiSys Workshop on Mobile Opportunistic Networking*, 2007, p. 74.

[3] Y. Lin, B. Li, and B. Liang, "Efficient network coded data transmissions in disruption tolerant networks," in *Proc. IEEE INFOCOM*, 2008, pp. 1508–1516.

[4] J. Widmer and J. Le Boudec, "Network coding for efficient communication in extreme networks," in *Proc. ACM SIGCOMM Workshop on Delay-Tolerant Networking*, 2005, vol. C, no. 5, p. 291.

[5] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *Proc. ACM SIGCOMM Workshop on Delay-Tolerant Networking*, 2005, vol. M, p. 236.

[6] J. Dong, R. Curtmola, R. Sethi, and C. Nita-Rotaru, "Toward secure network coding in wireless networks: threats and challenges," in *Proc. 4th Workshop on Secure Network Protocols*, Oct. 2008, pp. 33–38.

[7] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proc. 2nd ACM Conference on Wireless Network Security*, 2009, pp. 111–122.

[8] M. N. Krohn, M. J. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proc. 2004 IEEE Symposium on Security and Privacy*, 2004.

[9] L. Buttyán, L. Czap, and I. Vajda, "Securing coding based distributed storage in wireless sensor networks," in *Proc. IEEE Workshop on Wireless and Sensor Network Security (WSNS)*, 2008.

[10] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *Proc. IEEE INFOCOM*, 2007, pp. 616–624.

[11] D. Boneh, D. Freeman, J. Katz, and B. Waters, *Signing a Linear Subspace: Signature Schemes for Network Coding*. Springer, Mar. 2009, vol. 5443/2009, pp. 68–87.