Ad Hoc Networks 45 (2016) 80–103

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Impact of hash value truncation on ID anonymity in Wireless Sensor Networks

Ahmed Al-Riyami*, Ning Zhang, John Keane

School of Computer Science, The University of Manchester, Manchester, UK

ARTICLE INFO

Article history:
Received 17 October 2015
Revised 20 January 2016
Accepted 22 February 2016
Available online 22 March 2016

Keywords:
Hash value truncation
ID anonymity
Wireless Sensor Network

ABSTRACT

Hash functions have been used to address security requirements such as integrity, message authentication and non-repudiation. In WSNs, these functions are also used to preserve sensor nodes' identity (ID) anonymity, i.e., they are used to generate and verify dynamic pseudonyms that are used to identify sensor nodes in a communication session. In this latter application, there is an open issue as to how long the output of a hash function (i.e. hash value) we should use in pseudonym generation. The longer the hash value, the longer is the pseudonym, thus the harder it is to guess a pseudonym that is generated by using a hash function. On the other hand, the use of a longer hash value also means that the bandwidth and energy costs in transmitting the pseudonym will be higher. As sensor nodes typically have limited resources and are battery powered, the balance between the protection level of ID anonymity and performance and energy costs incurred in providing such a protection is an open issue. This paper investigates the use of hash value truncation in preserving ID anonymity in WSNs and the impact of hash value truncation on four criteria attributes (security against brute force attacks, probability of pseudonym collisions, energy trade-off and end-to-end packet delivery delay). It reports the possible impacts of other factors including the type and usage of hash functions, sensor node capabilities, adversary capabilities, ability to resolve pseudonym collisions, network density and data collection rate. The results show that the impacts of these factors may be contradictory. Therefore, the determination of an optimal level of hash value truncation should consider all trade-offs brought by these factors.

1. Introduction

Hash functions are computationally cheap to compute and hard to reverse, so they have been used to construct cryptographic algorithms or methods providing security services such as data integrity, origin authentication, entity authentication, anti-reply and non-repudiation. They are widely used in application areas such as virtual private networks (VPNs), secure electronic transaction, secure email, digital signatures, digital cash, electronic commerce, electronic voting and digital right management. As hash functions are computationally more efficient than other cryptographic primitives such as symmetric and asymmetric ciphers, they are also commonly used in security provisioning in resource-constrained networks such as Wireless Sensor Networks (WSNs). For example, one of the basic security services in WSNs where hash functions have also been applied is preserving the ID anonymity of sensor nodes.

Preserving node ID anonymity is a key element in providing security and privacy in WSNs due to its importance in encumbering the node capture attacks. Unlike the case in other wireless networks, sensor nodes in WSNs are prone to node capture attacks due to their unattended nature of deployment. Node capture attacks may result in data privacy compromise or further harm to the network operations. However, Becher et al. [1] have proved that such attacks are not as easy as they are assumed in literature. Adversaries need to invest time and effort to capture a node, obtain the stored data or modify the code within the node's memory, redeploy the captured node back to the network and start to mount further security and privacy attacks through the redeployed node. Therefore, adversaries often try to capture and compromise nodes that play a greater role in facilitating the network operations such as cluster heads or parent nodes located closer to the base station. Adversaries usually try to identify these nodes by analysing the communication relationship among the nodes through the study of the nodes' IDs carried in the exchanged packets. Preserving node ID anonymity is to try to hide such identi-

* Corresponding author. Tel.: +44 161 275 6270; fax: +44 161 275 6204.
E-mail addresses: ahmed.al-riyami@manchester.ac.uk, ahmed.alriyami@gmail.com (A. Al-Riyami), ning.zhang@manchester.ac.uk (N. Zhang), john.keane@manchester.ac.uk (J. Keane).

http://dx.doi.org/10.1016/j.adhoc.2016.02.019
1570-8705/

fying information from adversaries. One way of achieving this is through assigning dynamically changing identifiers (i.e. dynamic pseudonyms) to the communicating nodes in each transmitted packet.

A number of node ID anonymity schemes have been proposed in literature [2–6]. In these schemes, a communication node is assigned to, and identified by, one or more dynamic pseudonyms when it communicates with other nodes. Dynamic pseudonyms are usually generated using a hash function on a per message basis, i.e. the hash value produced from a hash function is used to construct such a pseudonym. Therefore, there is an issue as to how long a hash value we should use when generating a pseudonym. The longer the hash value, the longer the pseudonym, thus the stronger the protection the pseudonym may offer. However, the use of a longer hash value also means that the energy cost in transmitting the pseudonym will be higher. According to [7], the most energy-consuming task performed by a communication node is data transmission. Therefore, it is important to investigate the implications of using different hash value lengths for node ID anonymity preservation and on performance and energy costs, and what are the other factors that may influence the selection of hash value lengths in the context of ID anonymity preservation in WSNs.

This paper reports an investigation on the use of hash value truncation to preserve ID anonymity in WSNs and the impact of hash value truncation on the security and the performance and energy costs of the approach. The investigation is based on two existing ID anonymity schemes, the Efficient Anonymous Communication (EAC) scheme [5] and the Cryptographic Anonymous Scheme (CAS) [2]. The paper also reports the impacts on the trade-off of other factors including the type and usage of hash functions, sensor node capabilities, adversary capabilities, ability to resolve pseudonym collisions, network density and data collection rate. Although hash truncation has been proposed to reduce data transmissions in general [8–13], the analysis of its impact on ID anonymity and the costs incurred is lacking in the literature. The results from this study may be useful where pseudonym schemes are used to preserve ID anonymity in any resource-constrained network environment such as WSNs. The framework in this study may also be used to assess which level of truncation should be applied for a given level of ID anonymity protection, what the performance and energy costs are like for a given level of truncation, and what are the other factors that should be considered when deciding on the level of truncation to use in pseudonym generations.

The rest of the paper is organised as follows: Section 2 overviews cryptographic hash functions along with potential attacks that may be mounted on the functions; Section 3 describes two existing ID anonymity schemes, EAC and CAS, that are based on hash functions; Section 4 describes the system model and assumptions used in our study; Section 5 discusses the study methodology; Section 6 analyses the impact of hash truncation on the security of the ID anonymity schemes and Section 7 analyses its impact on the collision resistance property; Sections 8 and 9 investigate the impact on the energy consumption and the end-to-end packet delivery delays, respectively; Section 10 discusses lessons derived from the study; and finally, Section 11 concludes the paper.

2. Cryptographic hash functions

This section overviews cryptographic hash functions. It covers the types and properties of cryptographic hash functions, and security attacks that may be mounted on the hash functions. It also explains hash value truncation.

2.1. Types and properties

A hash function maps an input of an arbitrary finite bit-length to an output of a fixed bit-length using a noninvertible compression process [14]. The term noninvertible here means that it is computationally hard to reverse. The resulting output is called a hash value, a hash tag or a digest. Hash functions can be classified into two groups [14]: unkeyed and keyed (see Fig. 2.1). An Unkeyed Hash Function (UHF) uses an algorithm that accepts only one input (data) in the process of generating a hash value. A Keyed Hash Function (KHF) accepts a cryptographic key in addition to the data to be hashed as input. KHFs are often used for achieving message authentication where the values generated are termed Message Authentication Codes (MACs). The following subsections provide more details about each hash function type along with their properties.

2.1.1. Unkeyed Hash Functions (UHFs)

An n-bit UHF is denoted as: $\{0,1\}^* \rightarrow \{0,1\}^n$. The function processes an arbitrary finite length input message $\in \{0,1\}^*$ and returns a hash value $\in \{0,1\}^n$, where $n \geq 1$. For a data input $x \in \{0,1\}^*$, $H(x) = y$ represents the computation of the hash function H on the data input x and returns the hash value $y \in \{0,1\}^n$. UHFs have the following five properties [14]:

1. Compression: Maps a message of an arbitrary length to an n-bit output, i.e., H: $\{0,1\}^* \rightarrow \{0,1\}^n$.
2. Ease of computation: For any $x \in \{0,1\}^*$, it is easy to compute $H(x)$.
3. Preimage resistance (also known as one-wayness): Given a hash value $y \in \{0,1\}^n$ (for which no preimage is known), it is computationally infeasible to find an input x such that $H(x) = y$.
4. 2nd preimage resistance (also known as weak collision resistance): Given an input x, it is computationally infeasible to find a second input x' such that $H(x') = H(x)$.
5. Collision resistance (also known as strong collision resistance): It is computationally infeasible to find two different inputs, x and x', such that $H(x) = H(x')$.

If the UHF satisfies the first four properties, it is called a One-Way Hash Function (OWHF) and if it additionally satisfies the collision resistance property then it is called a Collision Resistance Hash Function (CRHF) [15]. If the input space of a CRHF is larger than that of the output, then the function is a many-to-one map function, which means that hash collisions are unavoidable (i.e., multiple inputs may result in the same hash value). However, finding collisions in CRHFs is computationally difficult.

2.1.2. Keyed Hash Functions (KHF)

A KHF is a function that compresses an input of arbitrary length into a fixed length hash value using a secondary input which is the secret key. More formally, a KHF is a function $H_K : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{R}$, where the key space $\mathcal{K} = \{0,1\}^k$, the message space $\mathcal{M} = \{0,1\}^*$ and the range $\mathcal{R} = \{0,1\}^n$ for some k, $n \geq 1$. An instance computation of a KHF is represented as $H_K(x) = y$ where the key $K \in \{0,1\}^k$, $x \in \{0,1\}^*$ and $y \in \{0,1\}^n$. KHFs have the following properties [14]:

1. Compression: H_K maps an input of arbitrary finite length to an output of fixed length (n bits), i.e., H_K: $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^n$.
2. Ease of computation: Given a secret key K, computing $H_K(x)$ for all $x \in \{0,1\}^*$ is easy.
3. Key non-recovery: It is computationally infeasible to recover the secret key K, given one or more input-hash pairs $(x_i, H_K(x_i))$ for that K.