MANCHESTER
1824
The University of Manchester

The University of Manchester Research

Impact of Hash Value Truncation on ID Anonymity in Wireless Sensor Networks

DOI:
10.1016/j.adhoc.2016.02.019

Document Version
Accepted author manuscript

Link to publication record in Manchester Research Explorer

Citation for published version (APA):
Al-Riyami, A., Zhang, N., & Keane, J. (2016). Impact of Hash Value Truncation on ID Anonymity in Wireless Sensor Networks. *Ad Hoc Networks*, *45*, 80-103. https://doi.org/10.1016/j.adhoc.2016.02.019

Published in:
Ad Hoc Networks

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.

OPEN ACCESS

Download date:25. Aug. 2022

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

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 capa-

*Corresponding author

Email addresses: ahmed.al-riyami@manchester.ac.uk (Ahmed Al-Riyami), ning.zhang@manchester.ac.uk (Ning Zhang), john.keane@manchester.ac.uk (John Keane)

bilities, 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.

Keywords: hash value truncation, ID anonymity, Wireless Sensor Network

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 identifying 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, 3, 4, 5, 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, 9, 10, 11, 12, 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 Figure 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.

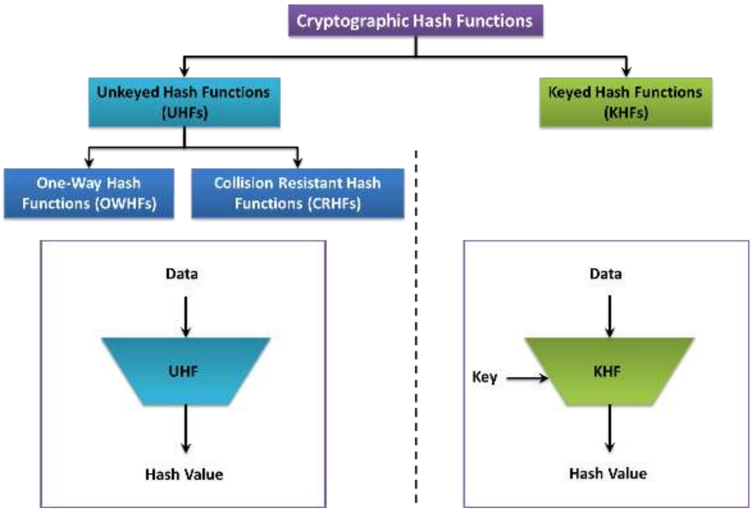

Cryptographic Hash Functions
Unkeyed Hash Functions (UHFs)
Keyed Hash Functions (KHFs)
One-Way Hash Functions (OWHFs)
Collision Resistant Hash Functions (CRHFs)
Data
UHF
Hash Value
Data
Key
KHF
Hash Value

Figure 2.1: Classification of hash functions.

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. 2^{nd} 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.
4. Computation resistance: Given zero or more input-hash pairs $(x_i, H_K(x_i))$, it is computationally infeasible to find an input-hash pair $(x, H_K(x))$ for any new input $x \neq x_i$ without knowing K.

2.2. Security Attacks on Hash Functions

There are two types of attacks on cryptographic hash functions: brute force and cryptanalysis [16]. Brute force attacks can be mounted on all hash functions regardless of their internal structure. The effectiveness of the brute force attack is dependent on the size of the hash value, and, in the case of KHFs, it is also dependent on the size of the cryptographic key used. Cryptanalysis attacks, on the other hand, target the internal structure of a hash function; more details on such attacks can be found in [17]. In the following analysis, we omit the cryptanalysis attacks, and focus only on brute force attacks. This is because cryptanalysis attacks concern the internal structure of a hash function, i.e. the design of a hash function. By selecting a more secure hash function, we can mitigate the risk imposed by this class of attacks. In addition, as our focus in this paper is on the impact of hash value truncation, we focus on attacks that are dependent on the length of a hash value.

2.2.1. Brute force attacks on UHFs

There are three means of launching brute force attacks targeting the main properties of UHFs:

1. Preimage attack (i.e., attacking the one wayness property): The target of this attack is to find a preimage value x that hashes to a given hash value $H(x)$. Given a hash function that produces a hash value of length n, it takes about 2^n evaluations of the hash function to find a preimage value [14].
2. 2^{nd} preimage attack (i.e., attacking the weak collision resistance property): The target of this attack is to find a 2^{nd} preimage value $x^{'}$ that will result in

the same hash value $H(x)$ where $x^{'} \neq x$ provided that the adversary have access to the pair $(x, H(x))$. Given a pair $(x, H(x))$ and a hash function that produces hash values of length n, it takes about 2^n evaluations of the hash function to find a 2^{nd} preimage value [14].

3. Birthday attack (i.e., attacking the strong collision resistance property): The target of this attack is to find collisions for a given hash function. It is much easier to find collisions than finding 2^{nd} preimages. This is due to the birthday paradox [18] which states that in a group of 23 people, the probability that there are at least two people with the same birthday is 50%. Based on the birthday paradox, for a given hash function that produces hash values of length n, it takes about $2^{n/2}$ evaluations of the hash function to find a collision [14].

2.2.2. Brute force attacks on KHFs

Attacking KHFs is more complicated than attacking UHFs, as a cryptographic key is used when generating a hash value from a KHF and an attacker may need to know input-hash value pairs (i.e. the pairs of input text, x_i, and its hash value, $H_K(x_i)$) to mount successful attacks. The following highlights the possible means of mounting brute force attacks against KHFs:

1. Preimage attack: In this attack, the adversary tries to find a preimage x for a given hash value $H_K(x)$ without prior knowledge of the key. The adversary tries to guess a value x and verifies if the resulting hash value of this input x equals to the given hash value. A successful guess is computationally very difficult without knowledge of the key. To verify the guesses without knowing the key, the adversary would need to interact with a party that can provide a hash value for a given input. In other words, the adversary would need to perform online verifications for each guessed value. Thus, the difficulty in successfully mounting such an attack is very much dependent on the nature of the applications. For example, some applications impose a limit on the number of hash operations an entity is allowed to perform, and in such a case the probability of successfully

mounting an attack is very low. In the worst case where an application allows an unlimited number of hash operations (online verifications) one entity may perform, the complexity of this attack is 2^n [19], where n is the hash value size.

2. 2^{nd} preimage attack: In this attack, the adversary knows the pair (x_1, $H_K(x_1)$) and tries to find another input x_2 such that $H_K(x_2) = H_K(x_1)$. Again the attack needs to be verified online and its complexity is 2^n [20].
3. Exhaustive key search attack: The aim of this attack is to deduce the KHF cryptographic key that is used in the computation of the hash values. An adversary may guess the key using exhaustive search. However, for this attack to succeed, the adversary requires one or more known input-hash pairs. Let the size of the key be k bits and the length of the resulting hash value be n bits, then the adversary tries to compute the KHF using all possible 2^k keys one by one and to validate the result against a given input-hash pair (offline validation). When $k > n$, there could be a chance that more than one key may result in the same hash value. For a KHF algorithm that approximates a random function, the expected number of matching keys equals 2^{k-n}. To further reduce the uncertainty, the adversary needs to do another round using another input-hash pair, but in this case only the matching keys from the previous round are checked. If there remains more than one matching key from the second round (the number of matching keys expected after the second round is 2^{k-2n}), then the adversary needs to check a third input-hash pair, and so on. This process repeats until the adversary finds a single matching key. The number of required input-hash pairs is k/n and the complexity of the attack is 2^k [19].
4. Random forgery attack: The aim of this attack is to guess the hash value for an arbitrary message. For a KHF function with n bits output and k bits key, there are two methods to perform this attack. The first is to randomly guess the hash value; the success probability of this method is 2^{-n}. The other method is to guess the key and then compute the hash

value based on this key; the success probability of this method is 2^{-k}. The attack can only be verified online and its complexity is $2^{min(n,k)}$ [21].

2.3. Hash Value Truncation

Hash value truncation has been used in the design of several standard hash functions. For example, SHA-224 is a truncated version of SHA-256, and SHA-384, SHA-512/224 and SHA-512/256 are truncated versions of SHA-512. Rules for hash value truncation have been defined in several hash and MAC-related standardisation documents, such as FIPS-180-4 [8], FIPS-198 [9], ISO/IEC 9797-1 [10], ISO/IEC 9797-2 [11], RFC 2104 [12] and NIST 800-38B [13]. The rules can generally be summarised as follows [22].

Let λ be the desired length in bits after truncation of an n-bit hash value. The truncated hash value may be used if the following requirements are met:

1. λ should be smaller than the full-length hash value, n (i.e., $\lambda < n$).
2. The λ left-most bits of the full-length hash value should be selected after truncation and the remaining rightmost bits are discarded.
3. If the collision resistance property is required, λ should be at least twice the required collision resistance strength c (in bits) for the truncated hash value (i.e., $\lambda \geq 2c$).

In the rest of this paper, we use λ to denote the length (i.e., number of bits) of a truncated hash value.

3. Use of Hash Functions for Node ID Anonymity in WSNs

One way to preserve node ID anonymity in WSNs is to use dynamic pseudonyms to identify nodes when they communicate. There are several methods proposed to generate and verify the dynamic pseudonyms, such as pseudonym sub-ranges [2], hash functions [2, 3, 4, 5, 6], onion routing and label switching [23], Phantom ID [24], Bloom Filters [25] and orthogonal code [26]. The most popular method is the use of hash functions owing to their attractive properties such as low computational costs. To understand the impact of hash truncation on the security and performance of ID anonymity schemes, we have selected two existing schemes that use hash functions to preserve node ID anonymity, the Efficient Anonymous Communication (EAC) scheme [5] and Cryptographic Anonymity Scheme (CAS) [2]. The reasons for selecting these two schemes are three-fold:

1. Both schemes provide a higher level of ID anonymity protection than other schemes as they anonymise node IDs in both unicast and broadcast communication modes.
2. Both schemes are described comprehensively in terms of the sizes and structures of the pseudonyms and the use of these pseudonyms in different exchanged messages. This allows for a more thorough analysis of the implications of hash truncation on the security and performance of the schemes.
3. EAC is based on UHF whereas CAS is based on KHF, allowing us to analyse the truncation of hash values generated using UHF and KHF in this problem context. The analysis results based on these two schemes would be generic and can be applied to any other ID anonymity schemes that are either based on UHFs or on KHFs.

The following subsections describe these two schemes.

3.1. Efficient Anonymous Communication (EAC)

The pseudonyms in EAC are generated using a UHF on a per message basis. Prior to mutual communication between any two nodes, both the sender and the receiver generate a pseudonym for the next message to be transmitted (*Next Message Pseudonym*) independently and store it in a pseudonym table in their memory, making it ready for future communication. The pseudonym is computed as: $\textit{Next Message Pseudonym} = H(\textit{Current Pseudonym} \oplus \alpha)$, where α is a secret known only by the sender and the receiver and $\oplus$ is the exclusive-or operator.

EAC works as follows. Before deploying the nodes, each node, i, is preloaded with a unique ID (ID_i), a pairwise key shared with the base station (k_i), a broadcast key (k_b^i), two random numbers (α_i and β_i) and two UHFs, H_1 and H_2. After the node deployment, EAC operations are carried out in two phases, a network setup phase and an operational phase.

3.1.1. Network Setup Phase

In the network setup phase, each node uses the hash function H_1 to generate two anonymous identities; one serves as the node's global anonymous identity AI_i (to be used for anonymous unicast communication with the base station) and the other as its anonymous broadcast identity BAI_i (to be used for anonymous local broadcasts to the node's neighbours). These identities are generated using Equation (3.1).

$$\begin{aligned} AI_i &= H_1(ID_i \oplus \alpha_i) \\ BAI_i &= H_1(ID_i \oplus \beta_i) \end{aligned} \tag{3.1}$$

Then, the node exchanges some parameter values with each of its neighbouring nodes using a one-hop broadcast message. The parameters include $ID_i, k_i, k_b^i, \alpha_i, \beta_i$ and $Hop_{i,BS}$, where $Hop_{i,BS}$ is the smallest hop count between node i and the base station. Upon receiving the broadcast messages from all its neighbours, node i uses the received parameter values to compute the following items for each of its neighbouring nodes, j:

1. A new random number $\alpha_{i \leftrightarrow j} = H_1(ID_i \oplus ID_j)$.
2. A pairwise key $k_{i \leftrightarrow j} = H_2(k_i + k_j + \alpha_i + \alpha_j)$.
3. A one hop anonymous identity $OHAI_{i \leftrightarrow j} = H_1(\alpha_i \oplus \alpha_j)$.
4. An anonymous acknowledgment identity $AAI_i = H_1(ID_i)$.
5. An anonymous broadcast identity for the neighbour $BAI_j = H_1(ID_j \oplus \beta_j)$.

Each node then uses the computed values to update its neighbour information table T_i as shown in Table 3.1. After updating the table, the node deletes its real ID and the one-hop broadcast messages received from other neighbours.

3.1.2. Operational Phase

When a node i is to send data D to the base station multi-hop away (say through node j), it composes a message as follows:

Table 3.1: The neighbour information table T_i at node i in EAC [5].

Anonymous broadcast identity	BAI_j	...
One hop anonymous identity	$OHAI_{i\leftrightarrow j}$	...
Anonymous acknowledgement identity	AAI_j	...
Shared random number	$\alpha_{i\leftrightarrow j}$	...
Shared broadcast random number	β_j	...
Shared broadcast key	k_b^i	...
Shared one-hop key	$k_{i\leftrightarrow j}$	...
Link direction	$linkdir_{i\to j}$	...

$M_{i\to j} = OHAI_{i\leftrightarrow j}||E_{k_{i\leftrightarrow j}}\big(AI_i||E_{k_i}(D)||H\big(AI_i||E_{k_i}(D)\big)\big)$, where || is a concatenation operator. After sending the message to node j, node i updates its global identity and the one hop anonymous identity using Equation (3.2).

$$\begin{aligned} AI_i &= H_1(AI_i \oplus \alpha_i) \\ OHAI_{i\leftrightarrow j} &= H_1(OHAI_{i\leftrightarrow j} \oplus \alpha_{i\leftrightarrow j}) \end{aligned} \tag{3.2}$$

When node j receives the message, it checks its table T_j for a matching $OHAI_{i\leftrightarrow j}$. If no match is found, node j drops the message. Otherwise, node j fetches the pairwise key $k_{i\leftrightarrow j}$ from T_j and uses the key to decrypt the received message and check the integrity by computing a digest of the payload, $AI_i||E_{k_i}(D)$, and comparing it with the received digest, $H(AI_i||E_{k_i}(D))$. Then, node j checks its routing table for the next hop in the route to the base station, say node r, and constructs a new message to node r as follows:

$M_{j\to r} = OHAI_{j\leftrightarrow r}||E_{k_{j\leftrightarrow r}}\big(AI_i||E_{k_i}(D)||H\big(AI_i||E_{k_i}(D)\big)\big)$

Node j also updates its pairwise identities with node i and r using Equation (3.3):

$$\begin{aligned} OHAI_{i\leftrightarrow j} &= H_1(OHAI_{i\leftrightarrow j} \oplus \alpha_{i\leftrightarrow j}) \\ OHAI_{j\leftrightarrow r} &= H_1(OHAI_{j\leftrightarrow r} \oplus \alpha_{j\leftrightarrow r}) \end{aligned} \tag{3.3}$$

Upon the receipt of the message $M_{j\to r}$, node r follows the similar procedure as the one performed by node j described above to forward the message to the base station. If node r is the base station, it decrypts the payload using $k_{j\to r}$

and checks the integrity of the message by computing a digest of the payload and comparing it to the received digest. If the integrity check is positive, the base station obtains AI_i and searches its T_r table for a match to identify the original sender of the data. It then uses the corresponding key, k_i, shared with node i to decrypt $E_{k_i}(D)$ and obtain D. The base station then updates the one hop anonymous identity shared with node j as well as the global anonymous identity AI_i shared with the original sender i using Equation (3.4).

$$\begin{aligned} OHAI_{j\leftrightarrow r} &= H_1(OHAI_{j\leftrightarrow r} \oplus \alpha_{j\leftrightarrow r}) \\ AI_i &= H_1(AI_i \oplus \alpha_i) \end{aligned} \tag{3.4}$$

In EAC, the sender and receiver are required to update their shared anonymous identities after each successful message transmission. However, due to channel impairments or transmission errors, message loss may occur causing the sender and receiver to be out of synchronization, thus hindering future message transmissions between the two nodes. To address this problem, EAC uses anonymous acknowledgements as follows: when a node i sends a message to another node j it appends its own acknowledgment identity AAI_i to the message as follows:

$$M_{i\rightarrow j} = D_{rand}||OHAI_{i\leftrightarrow j}||E_{k_{i\leftrightarrow j}}\big(AAI_i||AI_i||E_{k_i}(D)||H\big(AAI_i||AI_i||E_{k_i}(D)\big)\big)$$

Here D_{rand} is a random string padded to the message to make each sent message equal in size. Upon receiving $M_{i\rightarrow j}$, node j decrypts the message, obtains AAI_i and adds it as part of the message to be forwarded to the next node in the route, say node r, as follows:

$$M_{j\rightarrow r} = AAI_i||OHAI_{j\leftrightarrow r}||E_{k_{j\leftrightarrow r}}\big(AAI_j||AI_i||E_{k_i}(D)||H\big(AAI_j||AI_i||E_{k_i}(D)\big)\big)$$

When node i overhears the message $M_{j\rightarrow r}$ containing the same AAI_i, it knows that the message is received successfully and both the sender and receiver update their one-hop anonymous identity $OHAI_{i\leftrightarrow j}$. If node i does not overhear the message $M_{j\rightarrow r}$ containing the same AAI_i after a time out period, it assumes that the message is lost and retransmits $M_{i\rightarrow j}$ to node j. The final destination is required to send an explicit acknowledgment message back to its one-hop neighbour.

3.2. Cryptographic Anonymity Scheme (CAS)

The CAS scheme is designed to protect node ID anonymity in clustered WSNs. In CAS, the pseudonyms are generated using a KHF. Unlike EAC, the pseudonyms in CAS are not stored locally in each node's memory; rather they are computed by the sender before sending a message and verified by the receiver upon receiving the message. A pseudonym in the CAS scheme is computed as:

$Pseaudonym = Ind||H_K(a \oplus seq)$,

where Ind is an index value, K is a hash key, a is a random seed, both K and a are shared between a pair of communicating nodes and seq is a message sequence number carried in each message. There are also two phases in CAS: setup and operational as described below.

3.2.1. Network Setup Phase

During the setup phase, nodes exchange parameter values with their neighbours. The parameter values include keys, random seeds and message sequence numbers. Each node uses the parameter values to update a pseudonym table that contains an entry for each of its neighbouring nodes. In the operational phase, the stored parameter values are used to compute and verify pseudonyms for pairwise and broadcast communications. Table 3.2 shows an example of a pseudonym table.

Table 3.2: Pseudonym table for node u in CAS [2].

Index	**Mutual exchange**		**Neighbour's clustere exchange**			**Neighbour's clustere key**	**Neighbour's hash key**	**Shared key**	**Shared hash key**	**Neighbour's index**
	a	**Seq**	a	b	**Seq**					
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
Ind_u	a_{uv}	seq_{uv}	a_{cv}	b_{cv}	seq_{cv}	$k_{cv}^{'}$	K_{cv}	$k_{uv}^{'}$	K_{uv}	Ind_v
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.

3.2.2. Operational Phase

The anonymous communications in the operational phase of the CAS scheme are carried out as follows. A message from node u to node v with the base station as the final destination is composed as: $M_{uv} = SID||RID||EncryptedPayload||$ seq_{uv} , where SID is the end-to-end mutual pseudonym computed as $SID = Ind_v||H_{K_{Bu}}(a_{Bu} \oplus seq_{uv})$ and RID is the next-hop mutual pseudonym computed as $RID = Ind_v||H_{K_{uv}}(a_{uv} \oplus seq_{uv})$, Ind_v is the index in node v's pseudonym table where parameters related to node u are stored, K_{Bu} and a_{Bu} are the hash pairwise key and the random seed shared between node u and the base station, respectively, K_{uv} and a_{uv} are the hash pairwise key and the random seed shared between node u and node v, respectively and seq_{uv} is the current message sequence number for the mutual communication between nodes, u and v. When v receives the message M_{uv} , it retrieves Ind_v from the message and searches its pseudonym table for a match. If a match is found, node v uses the corresponding values of K_{uv} and a_{uv} from the table to compute $H_{K_{uv}}(a_{uv} \oplus seq_{uv})$. If the computed value equals the received value, then node v is assured that it is the intended receiver of the message. Otherwise it discards the message. When node v forwards the message to the base station, it includes the received sequence number seq_{uv} in the forwarded message. When the message eventually reaches the base station, the base station computes $H_{K_{Bu}}(a_{Bu} \oplus seq_{uv})$ for every node in the network. Then, it compares the computed hash value with the hash value received in the message to identify the original source of the message.

4. System Model and Assumptions

This section explains the system model and assumptions used in our investigation. The WSN consists of a large number (>500) of similar low-cost, resource-constrained static sensor nodes and a single resource-rich base station. The data collection model is a continuous model in which the sensors are used to periodically measure a physical phenomenon and send the collected measurements to the base station. The data is collected every minute. EAC uses a flat network topology where the data is delivered from a source node to the base station using probabilistic routing. CAS uses a hierarchical topology where the sensor nodes are organised into clusters. Each cluster has an elected cluster head. Data collected by sensor nodes are first sent to their respective cluster heads and the cluster heads then forward the data to the base station through other cluster heads. The network models of EAC and CAS are illustrated in Figure 4.1.

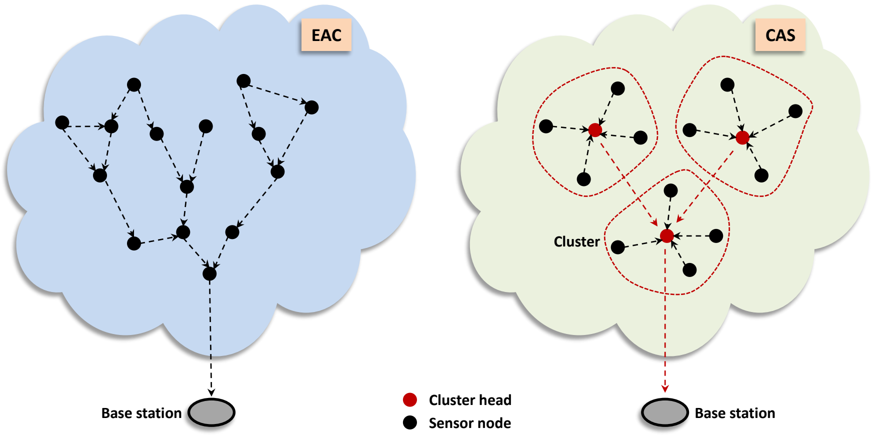


Figure 4.1: Network models of EAC and CAS.

An important parameter in our investigation is the network density. We use the size of a node's neighbourhood (i.e., the number of single-hop neighbours each node has) to control the value of this parameter. The size of the pseudonym table used by each node is proportional to the size of the node's neighbourhood. We set the network density value to three bands, low, medium and high, as defined below. The selections of these values are based on the connectivity analysis presented in [27, 28]:

1. Low density (the network has fewer than 30 nodes).
2. Medium density (the network has between 30 and 70 nodes).
3. High density (the network has more than 70 nodes).

To further scope our investigation, the following assumptions have been used:

1. All the analyses are performed based on the capabilities of Crossbow TelosB sensors [29]. The hardware specifications of TelosB are summarised in Table 4.1.
2. EAC assumes the use of MD5 for the generation of the pseudonyms and message digests, and AES (128 bits keys) for message encryption/decryption, whereas CAS uses CBC-MAC with Skipjack (80 bits keys) as the underlying block cipher for the generation of pseudonyms and the same block cipher (Skipjack) for message encryption/decryption. The security level of these cryptographic primitives may have been sufficient at the time when the papers were published. However, the collision resistance of MD5 was broken in 2013 [30] and Skipjack is no longer recommended by the National Institute of Standards and Technology (NIST) as the skipjack security limit was set to expire in 2010 [31]. Therefore, to make our analysis results up-to-date, we assume the use of SHA1 instead of MD5 in EAC and HMAC-SHA1 instead of CBC-MAC in CAS for the generation of pseudonyms. We also assume the use of AES (128 bits keys) for message encryption/decryption in both schemes. It is worth mentioning that these assumptions will not affect the length of the hash values suggested in the original papers, i.e., 128 bits for EAC and 64 bits for CAS; they only affect the execution times of the operations such as pseudonym generation and verification, message integrity token generation and verification and message encryption and decryption, etc. A benchmark on the execution times of these cryptographic primitives on a TelosB mote is summarised in Table 4.2.
3. Without loss of generality, it is assumed that the hash functions used are ideal, i.e., they possess all the properties mentioned in Section 2.1 and cannot be broken by an effort less than the brute force attacks. The hash values approximate a uniform random variable (i.e., the probability of producing a specific hash value = 2^{-n} where n is the length of the hash

value).

Table 4.1: Hardware specifications of TelosB mote.

HW components	**Specification**
Microcontroller	8 MHz TI MSP430
Program flash memory	48KB
RAM	10KB
External EEPROM	1 MB
Radio transceiver	Chipcon CC2420, 2.4 GHz , 250 kbps

Table 4.2: Execution time benchmarks for cryptographic algorithms in TelosB mote.

Cryptographic algorithms	**Operations**	**Execution times (ms)**
SHA1	Hash value generation (1 block)	4.64 [32]
HMAC-SHA1	MAC generation (1 block -128 bits key)	14.84 [32]
AES-CBC	Key expansion (128 bits key)	3.58 [33]
	Encryption (1 block)	3.77 [33]
	Decryption (1 block)	43.20 [33]

An adversary is assumed to have the following capabilities:

- Eavesdrop on communication channels.
- Launch offline brute force computations using a machine that is more powerful than a sensor node, e.g., a laptop. To benchmark the execution times of SHA1 and HMAC-SHA1, we have implemented the codes for both cryptographic algorithms using the cryptographic library, Crypto++ v.5.6.2, running on a machine with intel i7, 3.4GHz processor and 8 GB RAM. The codes were compiled using the 32-bit C/C++ Optimizing Compiler version 16.00.40219.01. Each program was run for 100,000,000 hash evaluations and the average time was calculated. The resulting execution times are given in Table 4.3.
- Send messages to a WSN node to launch online brute force forgery attacks utilizing the full channel capacity for this purpose. For a 250 kbps channel

capacity, the adversary can at most send around 470 packets/s where the average size of each packet is 68-bytes [34].

In this investigation, we have not considered the possibility that adversaries may be able to physically compromise a node. This is because if an adversary can compromise a node, then the adversary can gain access to all the secrets stored in the memory of the node including cryptographic keys, thus identifying the real ID of a sensor node. Breaking an anonymity scheme using such physical attacks is independent of the length of a pseudonym, or the hash value length, used. As our focus is on the implication of hash value lengths used, physical attacks on sensor nodes are outside the scope of this investigation.

Table 4.3: Execution times of cryptographic algorithms as measured on a machine with intel i7, 3.4GHz processor and 8 GB RAM.

Cryptographic algorithms	**Execution times (ns)**
SHA1	245.50
HMAC-SHA1	938.20

5. Analysis Methodology

As mentioned earlier, hash truncation can lead to reductions in bandwidth and energy costs when hash functions are used to generate pseudonyms to achieve node ID anonymity in WSNs. It is also well-known that truncating a hash value can have negative implications on security protection levels when hash functions are used as part of the measures to protect message integrity, origin authentication and non-repudiation [22]. For example, by truncating a n-bit hash value to a λ-bit for a certain hash function, the expected effort to find collisions is reduced from $2^{n/2}$ to $2^{\lambda/2}$. This means that it is easier to forge a hash value by using a different input and the security protection level is reduced.

However, the effects of hash value truncation on the ID anonymity protection level and on the performance and energy costs in our problem context are more complex than in the case where hash functions are used to achieve other security properties. Here we identify and discuss the factors to consider when selecting a hash value length in an ID anonymity protection scheme such as EAC and CAS.

As mentioned above, the shorter the hash value, the shorter the pseudonyms, thus the shorter the messages transmitted in an ID anonymity protection scheme. However, the shorter the hash value generated from a hash function, the weaker is the one-wayness of the hash function. In other words, if we use hash value truncation, it would be easier for an attacker to guess an input, x, given a hash value generated by the hash function, $H(x)$. For example, if x is a node's real ID and $H(ID)$ is a pseudonym used by this node, then it would be easier for an attacker to guess the real ID knowing the pseudonym. Therefore it is necessary to impose a minimum length for a hash value to ensure sufficient protection against the one-wayness attack and any other security attack that may be affected by the length of the hash value. Let us use λ_{SEC} to denote the minimum accepted truncated hash value length satisfying this security requirement.

In addition, the shorter the hash value used in pseudonym generations, the higher the probability of having pseudonym collisions in a pseudonym table of a receiving node. If there is a collision, the receiving node will need to resolve the collision, or to work out the sender's ID in the presence of the collision. Otherwise, this will hinder the operations of the ID anonymity scheme. In other words, we should also specify a minimum required truncated hash value length to control the probability of having pseudonym collisions under an acceptable threshold level. Let λ_{CR} denote this minimum required truncated hash value length satisfying this collision resistance requirement.

Furthermore, if a receiving node has another method to identify the transmitting node's ID in the event of a pseudonym collision in its pseudonym table, then the collision resistance property may not be an overriding requirement. However, to identify the transmitting node's ID in the presence of pseudonym collisions, the receiving node will need to perform some additional computations such as verifying the MAC value in addition to the pseudonym verification for all the colliding pseudonyms. In other words, the shorter the hash value we use in the pseudonym generations, the shorter the pseudonyms that will be generated, and the less the transmission costs incurred to the sender, but, at the same time, the higher the probability of receiving nodes having pseudonym collisions. The more the collisions, the more computations a receiving node will have to perform to identify the transmitting node. As both data transmissions and computations consume energy, there is a trade-off, indicating that there may be an optimal truncated hash value length that could lead to a minimum energy consumption. It is worth noting that the trade-off between achieving low transmission costs and achieving low pseudonym collision probability is actually the trade-off between the cost imposed on a transmitting node and the cost imposed on a receiving node. Let λ_E denote the optimal truncated hash value length which gives the lowest combined energy cost as caused by pseudonym transmissions by the transmitting node and collision resolution by the receiving node.

In addition to the energy cost in resolving collided pseudonyms, the receiving node will also need to spend more time to identify the real ID of the sender that has sent the incoming packet containing a collided pseudonym. Considering the multi-hop nature of a WSN, the delay introduced at each intermediate node due to collisions may affect the total end-to-end packet delivery delay. A packet delivery from a source node to the base station is bounded by the data collection period, i.e., the period between two successive data collections. The additional delay caused by hash value truncation should not make the maximum end-to-end packet delivery delay longer than the data collection period, causing interference between packets sent during two consecutive data collections. In other words, the truncated hash value should have a sufficient length such that the maximum end-to-end packet delivery delay stays within the data collection period, and let λ_D denote the minimum length of truncated hash values satisfying this requirement.

Figure 5.1 summarises the analysis methodology we use to investigate the lengths of truncated hash values against the requirements stated above. Each of the four truncated hash value lengths (i.e., λ_{SEC}, λ_{CR}, λ_E and λ_D) is investigated independently and then, based on the obtained results, the recommended truncated hash value length, λ_{REC} will be selected as shown in Equation (5.1).

$$\lambda_{REC} = \begin{cases} max(\lambda_{SEC}, \lambda_{CR}) & \text{if collision resistance is required} \\ max(\lambda_{SEC}, \lambda_E, \lambda_D) & \text{if collision resistance is not required} \end{cases} \tag{5.1}$$

To distinguish the notation used for different ID anonymity schemes, λ is superscripted by the respective scheme name, e.g., λ_{SEC}^{EAC} is used to refer to λ_{SEC} used in the EAC scheme.

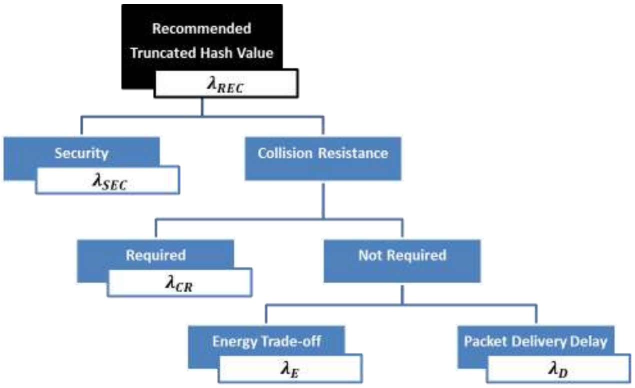


Figure 5.1: Analysis methodology.

6. Impact of Hash Truncation on Security

Brute force attacks on hash functions have been discussed in Section 2.2. Table 6.1 summarises the features of these attacks. This section analyses the implications of hash value truncation on the security of the EAC and CAS schemes in the presence of these brute force attacks.

The impact of the attacks is investigated using the following approach. Firstly, we identify attack methods or types, and for each attack type, we assess attack outcome to see if there is any motivation for an attacker to launch such

Table 6.1: Summary of brute force attacks against hash functions, where n is the length of a hash value, and k is the length of the crypto key.

Types of hash functions		**Brute force attacks**			
		Attacks	**Given**	**Success probability**	**Attack verification**
UHF	OWHF	Preimage	Hash value, $H(x)$	2^{-n}	Offline
		2nd preimage	Input and hash value pair, $(x, H(x))$	2^{-n}	Offline
	CRHF	Birthday attack	-	$2^{-n/2}$	Offline
KHF		Preimage	$H(x)$	2^{-n}	Online
		2nd preimage	$(x, H(x))$	2^{-n}	Online
		Exhaustive key search	$k/n \times (x, H(x))$ pairs	2^{-k}	Offline
		Random forgery	-	$2^{max(-n,-k)}$†	Online

† The max function indicates that the success probability is dependent on which guessing method (i.e., guessing the input to the hash function or the crypto key) is computationally cheaper.

an attack. If the outcome of the attack is useful to the attacker, we will proceed to analyse the attack feasibility and then address complexity. Otherwise, we will exclude the attack from our further analysis. The aim is to find the minimum length of hash values that could provide an acceptable level of protection against brute force attacks. The most important concepts to be used in our analysis are:

1. *Attack motivation*: The attack motivation refers to the case where the outcome of an attack justifies the investments (time, resources and efforts) made by an adversary in mounting the attack. Some of the brute force attacks mounted on a hash function may not bring any benefit to the adversary when the hash function is used by an application [35]. In such cases, there is little motivation for an adversary to invest on mounting such attacks that would not break or harm the application even though it is feasible to mount the attack on the hash function used by that application.
2. *Attack feasibility*: We say an attack is feasible if it is theoretically possible to successfully perform the attack. This assessment is based on the minimum information required to perform such an attack and the information

available to an adversary prior to mounting the attack.

3. *Attack complexity*: This is assessed in view of the resources and the timeframe available to an adversary to mount an attack. The complexity is first examined using full-length hash values and then using different truncated hash values. An attack timeframe available to an adversary depends on the nature of the attack. To scope our investigation, we specify an upper bound for each of the three types of timeframes as follows:
   a. The lifetime of a sensor node (T_N): The lifetime of a sensor node is constrained by the energy available to the node. According to [36], the lifetime of a typical battery-powered sensor node is bounded by the battery shelf life which is usually 10 -15 years. However, if a node uses an ambient energy source such as solar energy coupled with efficient energy storage devices, then its lifetime can be extended to more than 40 years [37]. Therefore, we assume $T_N = 50$ years as the maximum lifetime of a sensor node.
   b. The lifetime of a cryptographic key used in a KHF (T_K): We assume that a cryptographic key does not expire within the lifetime of the node that uses the key. Therefore, $T_K = T_N = 50$ years.
   c. The data collection period (T_{DC}): In our network model, it is assumed that $T_{DC} = 1$ minute.

6.1. EAC

As indicated in Section 3.1, a pseudonym in EAC is constructed as $H_1($ $OHAI_{i \leftrightarrow j} \oplus \alpha_{i \leftrightarrow j})$, where $OHAI_{i \leftrightarrow j}$ is the previously used pseudonym and $\alpha_{i \leftrightarrow j}$ is a secret only known to nodes, i and j. Brute force attacks on UHF include preimage, 2^{nd} preimage and birthday attacks. In addition, as part of the input to the hash function is secret (i.e., $\alpha_{i \leftrightarrow j}$), an adversary may attempt to forge the pseudonyms using a brute force method. Pseudonym forgery attacks include replay attacks and random forgery. In the following, we examine these attacks in detail.

Preimage Attack:

A potential generic attack on UHF, as discussed in Section 2.2.1, is to attack the one wayness property by trying to find a preimage for a given hash value (or pseudonym). Such an attack is useful to the adversary as finding and knowing the preimages may help the adversary to link different exchanged messages and further discover communication relationships among different nodes. For example, an adversary may get hold of one of the pseudonyms, e.g., $OHAI_{i \leftrightarrow j}$, by listening to the channel. The adversary may be able to reverse the hash value and check it against other intercepted pseudonyms and/or information learnt from other pseudonym messages. In this way, more information, such as the communication relationships among nodes, may be acquired by the adversary. However, if the length of the hash value is large enough, then the risk of this attack will be negligible. For example, given a hash value length of 128-bits, an adversary would need to perform 2^{127} offline hash evaluations on average for a success probability of 50%, and performing 2^{127} hash function evaluations will take around 1.33×10^{24} years based on the assumed SHA1 execution time presented in Table 4.3.

If the hash value is truncated to λ, where $\lambda < 128$ bits, then finding a preimage will be easier, i.e. take less time. How much easier is dependent on the length of λ. However, with the EAC scheme, the preimage value is $(OHAI_{i \leftrightarrow j} \oplus \alpha_{i \leftrightarrow j})$, where $\alpha_{i \leftrightarrow j}$ is a secret not known by the adversary. The length of $\alpha_{i \leftrightarrow j}$, as suggested by the authors, is 128 bits, which makes it hard for the adversary to discover the linkability between the exchanged messages. Additionally, as the length of the input is larger than that of the output, collisions are unavoidable. These collisions can actually make the attack more difficult to succeed, as collisions increase ambiguity in finding the preimage, and resolving the ambiguity requires more time, which may compensate for the reduction in the attack difficulty level caused by hash value truncation. Given λ, the number of expected collisions per pseudonym is $128/\lambda$.

2$^{\text{nd}}$ Preimage Attack:

Finding a 2$^{\text{nd}}$ preimage is another attack that may be mounted on UHF. However, in terms of compromising sensor node ID anonymity, this type of attack does not present any attack motivation. This is because this attack will not lead to the discovery of any useful information which may lead to the discovery of a sensor node real ID. 2$^{\text{nd}}$ preimages are normally used to break message integrity by trying to find another message that could hash to the same digest. In addition, finding a 2$^{\text{nd}}$ preimage means having a hash value collision, which increases ambiguity from the adversary point of view as the adversary's true intention is to find a distinct preimage to break the node ID anonymity. Hence, we do not investigate this attack further.

Birthday Attack:

With the same arguments as the ones made for the 2$^{\text{nd}}$ preimage attack above, the birthday attack does not present any attack motivation. Therefore, we do not investigate this attack further.

Replay attack:

A replay attack is aimed to lure a receiver to accept a replayed message by using a forged pseudonym. Here is a typical procedure of a replay attack: an adversary overhears a message $M_{i\rightarrow j} = OHAI_{i\leftrightarrow j}||E_{k_{i\leftrightarrow j}}\big(AI_i||E_{k_i}(D)||H\big(AI_i||$ $E_{k_i}(D)\big)\big)$, then tries to construct a new message, $M^{'}_{i\rightarrow j} = OHAI^{'}_{i\leftrightarrow j}||E_{k_{i\leftrightarrow j}}\big(AI_i$ $||E_{k_i}(D)||H\big(AI_i||E_{k_i}(D)\big)\big)$, where $OHAI^{'}_{i\leftrightarrow j}$ is a forged pseudonym, and sends this newly constructed message to the receiver. The receiver (i.e., node j) will accept the replayed message if and only if $OHAI^{'}_{i\leftrightarrow j}$ matches with node i's next-message pseudonym stored in the receiver's pseudonym table. If a match is found, the receiver will decipher the message using the corresponding key, $k_{i\leftrightarrow j}$, and the replay attack will be successful.

There are two methods to successfully forge the pseudonym. One is to directly obtain the next-message pseudonym stored in the receiver's pseudonym table which means that the adversary will need to access the pseudonym table managed by the receiver. It is assumed that this pseudonym table is hidden from the adversary, as, to get hold of this table, the adversary would need to physically compromise this node, interrupting the data collection process. The other method is to perform online guesses. Performing online guesses means that the adversary will need to perform verifications of his/her pseudonym guesses online and complete the verifications within a data collection period, T_{DC}.

Online pseudonym guesses can be done by either guessing a value of $\alpha_{i\leftrightarrow j}$ and computing the pseudonym using the UHF, or by directly guessing the pseudonym value. The probability of mounting this attack successfully is $2^{max(-l_\alpha,-n)}$, where l_α is the length of $\alpha_{i\leftrightarrow j}$ in bits and the max function indicates that the success probability is dependent on which of the two pseudonym guessing methods is computationally cheaper. As in this case $l_\alpha = n = 128$, the success probability of the attack is 2^{-128}. This means that the adversary will need to perform around 2^{127} online trials on average before hitting a match. This will take around 1.15×10^{28} years with the channel capacity of 250kbps. Obviously, this far exceeds the specified T_{DC}.

Reducing the pseudonym length will reduce the success probability to $2^{-\lambda}$ as $\lambda < l_\alpha$. Table 6.2 shows the average time required for the adversary to succeed in this attack given different values of λ. It can be seen that a length of 16 bits provides an attack time that is in the range of the data collection period. So, selecting one level higher, i.e., $\lambda = 24$ bits is sufficient to thwart the replay attack in our context (see highlighted cell in the table).

Random forgery attack:

This attack aims at making one or more of the nodes in a bounded geographical area to accept an illegitimate message. In mounting this attack, an adversary will construct a forged message with a forged pseudonym and a forged payload. To forge a payload, the adversary may use a key of his/her choice to encrypt some arbitrary data and then broadcast the forged message in a neighbourhood of b nodes hoping that one of the recipient nodes will find a matching pseudonym in its pseudonym table and will, therefore, accept the

Table 6.2: Time required by the adversary to succeed in the replay attack in EAC.

λ **(bits)**	**Time taken to succeed in the replay attack**
120	4.48×10^{25} years
112	1.75×10^{23} years
104	6.84×10^{20} years
96	2.67×10^{18} years
88	1.04×10^{16} years
80	4.08×10^{13} years
72	1.59×10^{11} years
64	6.22×10^{8} years
56	2.43×10^{6} years
48	9.50×10^{3} years
40	37.09 years
32	52.88 days
24	4.96 hours
16	1.16 minutes
8	0.27 seconds

message. According to this setting, each recipient node will have an average of b entries in its pseudonym table as the size of the pseudonym is proportional to the size of the neighbourhood. Therefore, the success probability of at least one node accepting the forged pseudonym is $(b^2 \times 2^{-n})$, where n is the length of the pseudonym. Again this attack can only be verified online and it has to be completed within the data collection period, T_{DC}.

However, accepting a forged pseudonym does not mean that the attack is successful as the recipient will still need to decipher the message and check its integrity as shown in Figure 6.1. As the adversary uses a key of his/her choice, the probability of the recipient accepting the message after accepting the pseudonym is around $2^{max(-k,-m)}$, where k is the length of the cryptographic key in bits and m is the length of the message digest. As k is normally larger than m, the probability of launching this attack successfully is $(b^2 \times 2^{-(n+m)})$. The use of a truncated hash value of λ bits length will increase the success probability to $(b^2 \times 2^{-(\lambda+m)})$. Based on this, selecting a larger value of m can offset the risk of this attack with a smaller value of λ. For example, when $b = 100$ (i.e., taking the highest network density under our assumption), $m = 32$ and $\lambda = 8$ (i.e., the shortest pseudonym length), the adversary will need to perform an average of 5.50×10^7 online trials to succeed in the attack. This will take around 1.35 days, which is beyond T_{DC}.

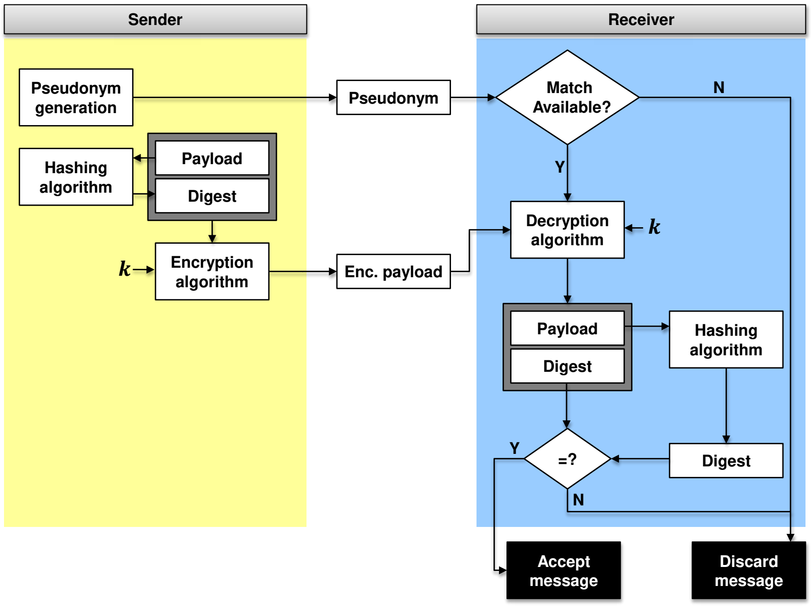


Figure 6.1: Steps required by a receiving node to accept a message in EAC.

From the above, it can be concluded that $\lambda_{SEC}^{EAC} = 24$ bits is sufficient to protect the scheme against brute force attacks. Further reduction in the length of the hash value makes EAC susceptible to replay attacks.

6.2. CAS

CAS uses pseudonyms of the form $Ind||H_K(a \oplus seq)$. The index length is 16 bits, the hash value length $n = 64$ bits and the key length $k = 128$ bits. The specification of the CAS scheme does not include a message authentication code (MAC) to protect the message authenticity and integrity. However, if message authenticity and integrity are not protected, an adversary can easily launch attacks targeting at these properties. So, in the analysis detailed in this section, we assume that every message is also integrity-protected using a MAC value. Brute force attacks on KHF are the preimage, 2^{nd} preimage, exhaustive key search and random forgery attacks. Similar to the case of EAC, the 2^{nd} preimage attack does not pass the motivation assessment, therefore we do not analyse it further.

Preimage attack:

The preimage attack in CAS may reveal the secret, a, which should only be known by the communicating node pair. Knowing a may expose any linkage of pseudonyms used by different messages leading to the compromise of the sender and receiver ID anonymity. To launch the attack, an adversary may try to guess the input value to the hash function using online verifications. The online verifications may continue until a guess proves successful, so the attack timeframe is limited to the lifetime of the node, T_N.

With this attack on a CAS pseudonym, the adversary has to increment the sequence number for each such attempt to pass the anti-replay check by the receiver. In addition, the receiver will also perform MAC verification before accepting the message. If the verification outcome is negative, the message will be dropped (see Figure 6.2). If the length of the pseudonym hash value is 64 bits (i.e. $n = 64$ bits) and the MAC value length (m) is 32 bits, then, the number of online verifications is 2^{95} to mount such an attack with a success probability of 50%. This will take 2.67×10^{18} years which is beyond the lifetime of a sensor node. Truncating the hash values will reduce this effort. Table 6.3 shows the time required to mount a successful attack given varying λ values. From the table, it can be seen that, using $\lambda = 16$ bits should be sufficient to thwart the risk of this attack.

Exhaustive key search attack:

The exhaustive key search attack requires known input-hash pairs and is carried out offline as discussed in Section 2.2.2. The minimum number of re-

Table 6.3: Hash value length vs. time taken to mount a successful preimage attack in CAS.

λ (bits)	Time taken to mount a successful attack with probability of 50% (years)
56	1.04×10^{16}
48	4.08×10^{13}
40	1.59×10^{11}
32	6.22×10^{8}
24	2.43×10^{6}
16	9.50×10^{3}
8	37.09

quired input-hash pairs is 2 based on the given sizes of n and k. To analyse the feasibility of the exhaustive key search attack on CAS, we further assume that a CAS pseudonym and the MAC value attached to each CAS message are generated using the same KHF, but they may use the same key or different keys, i.e.,

Case 1: The pseudonym and the MAC are generated using their respective (unrelated) keys.

Case 2: The pseudonym and the MAC are generated using the same (or related) key.

For Case 1, the adversary requires 2 input-hash pairs before attempting offline computation to recover the key. As the input text to the hash function is unknown to the adversary, he/she will have to eavesdrop the channel to intercept pseudonyms and try to find the input text by launching a preimage attack on the intercepted pseudonyms. Launching the preimage attack using online verification is very hard when using the full length of the hash value as discussed earlier. Recovering the MAC key does not help the adversary either as the pseudonym generation uses a different (unrelated) key.

For Case 2, as both pseudonyms and MAC values are generated using the same key, if an adversary could recover the key from a MAC value, then the attack is successful. To examine this case, we further discuss two scenarios, (2.a) the size of the MAC value, m, equals to that of the cryptographic key used, i.e., $m = k$, and (2.b) the size of the MAC value, m, is less than that of the key, i.e., $m < k$. For (2.a), the adversary will only need one input-MAC pair to perform this attack. Acquiring a single input-MAC pair through eavesdropping the channel is not a difficult task. For scenario (2.b), the adversary requires more than one input-MAC pairs to recover the key. To acquire more input-MAC pairs, the adversary will need to intercept more exchanged messages. In addition, the adversary would also need to identify the links among the intercepted messages to discover if multiple MACs are actually computed using the same key. In other words, among the intercepted messages, the adversary needs to identify those that are exchanged between the same pair of communicating nodes. This may be done by launching preimage attacks on the pseudonyms carried in the intercepted messages.

From the above analysis, it can be seen that to launch the exhaustive key search attack, the adversary needs to launch the preimage attack except in Case 2.a. For Case 2.a, if the adversary could get hold of an input-MAC pair, he/she would still need to recover the key by offline computing an average of 2^{k-1} KHF computations with a success probability of 50%. To attempt 2^{127} number of KHF computations, the adversary will be successful in 5.062×10^{24} years with the given computing capability. It is worth noting that the attack timeframe is bounded by the lifetime of the key which is assumed to be the same as the lifetime of the node. The shorter the key, the easier the attack, and a successful attack may lead to the compromise of the anonymity scheme.

It is also worth noting that if hash truncation is applied (say when $\lambda = 8$ bits), launching preimage attacks may become feasible, thus exhaustive key search attacks detailed in Case 1 and 2.b also become feasible at this level of truncation. However, the adversary will require more input-hash pairs to successfully launch the key search offline and even if the adversary can succeed in obtaining all the required input-hash pairs, the use of a large key size will eliminate the risk of this attack. For key size recommendations, the reader is referred to [38].

Random forgery attack:

In this attack, an adversary forges a message and broadcasts it to a neighbourhood of a number of nodes hoping that one of the nodes will accept the message. Each CAS message contains a sequence number to protect it against replay attacks. So to succeed in launching such a random forgery attack, the adversary would have to (a) select the correct index value, (b) use a sequence number that is larger than the current value (in order to pass the anti-replay check by the receiving node), (c) guess the correct pseudonym (i.e., hash value), (d) forge a payload and (e) forge a MAC that could pass the integrity check. This attack is bounded by the sensor node lifetime, T_N.

For (a), assuming that the indexes listed in the pseudonym table of a receiving node are unique and incremental, the adversary may select a small index value to guarantee that all or most of the receiving nodes will find a match. For (b), the adversary may choose a large message sequence number so that the message can pass the anti-replay check. For (c), the adversary may blindly guess a pseudonym value. For (d), the adversary may encrypt some arbitrary data using a random key or just use a random payload, and for (e), there are also two options, either computing the MAC value using a random key or using a random MAC value.

For the receiver to accept the pseudonym carried in a message, the outcomes of all the verifications have to be positive. Say the adversary broadcasts the forged message in a neighbourhood of b nodes. Upon the receipt of this forged message, we assume that all the receiving nodes have positively confirmed the first two verifications, (a) and (b). Then, each receiving node performs verification (c) by computing a hash value using the KHF and checking the result against the received hash value. The probability of passing this verification is 2^{-n}, where n is the length of the pseudonym hash value. After accepting the pseudonym, the node will proceed to verify the MAC value and decrypt the payload as shown in Figure 6.2. The probability of passing the MAC verification depends on the length of the MAC value (m), and this probability is 2^{-m}. So, based on the above analysis, the probability of successfully launching a random forgery attack in CAS is at most $b \times 2^{-(n+m)}$. If verifications (a) and (b) are taken into account, this probability would be lower. Table 6.4 shows the random forgery success probability for $n = 64$ bits, different MAC sizes and different network densities. It is clear from the table that the probability of success is extremely low even when the MAC value is small and the network density is large.

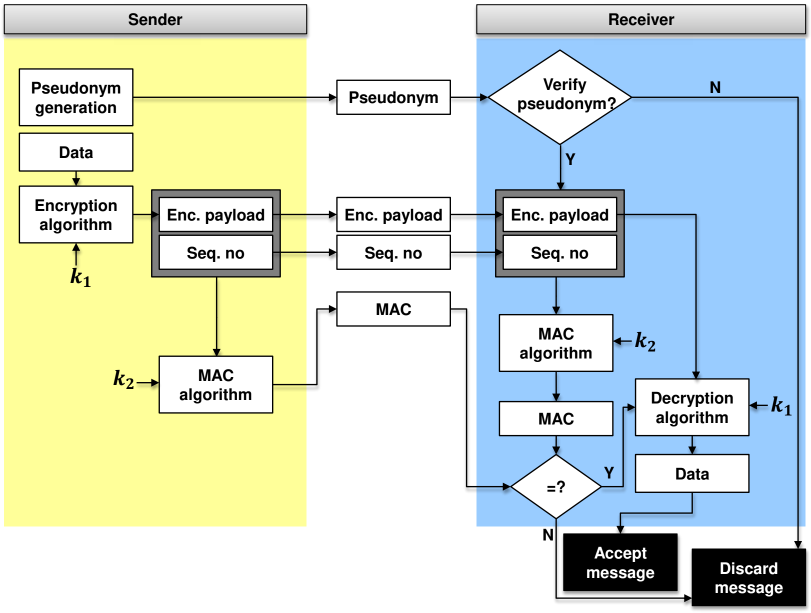
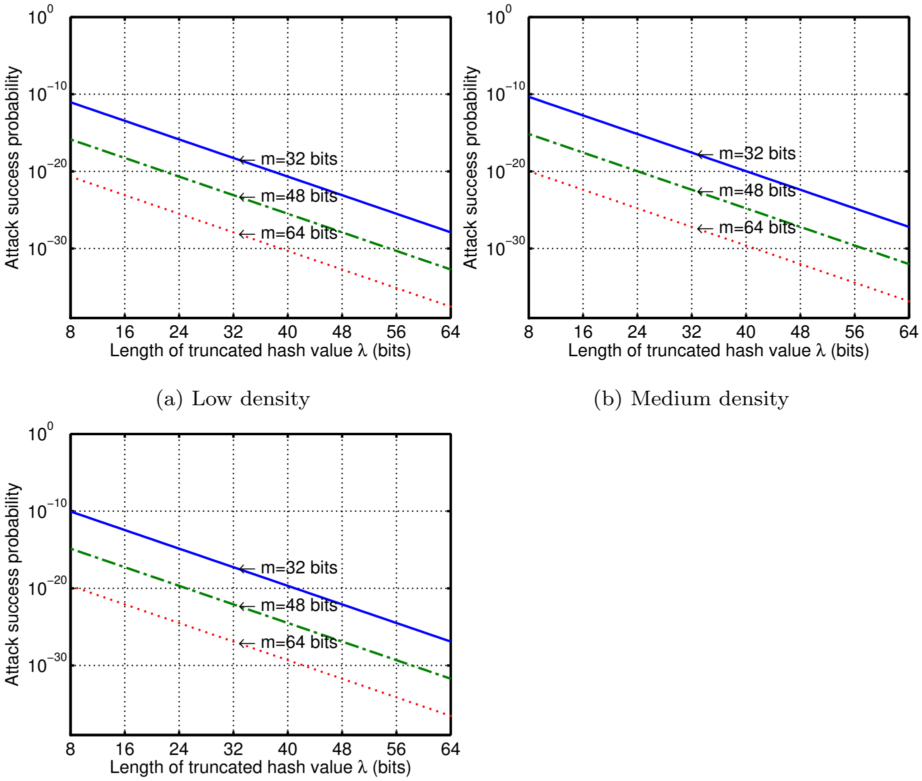


Figure 6.2: Steps required by a receiving node to accept a message in CAS.

Now, let us analyse the implications of hash truncation on the random forgery attack. Using a λ-bit hash value for a CAS pseudonym increases the success probability of the attack to $b \times 2^{-(\lambda+m)}$. Figure 6.3 shows the success probability of this attack against varying degrees of hash value truncations with three sizes of the MAC value, i.e. m = 64-, 48- and 32-bits and three network density levels, low, medium and high.

As can be seen in Figure 6.3, the probability for successful forgery attacks



(a) Low density

(b) Medium density

(c) High density

Figure 6.3: Random forgery attack success probability vs. different λ values and network density levels in CAS.

Table 6.4: Random forgery attack success probability in CAS for different MAC sizes.

MAC size m **(bits)**	**Success probability**		
	Low density (b=10)	**Medium density (b=50)**	**High Density (b=100)**
64	2.94×10^{-38}	1.47×10^{-37}	2.94×10^{-37}
56	7.52×10^{-36}	3.76×10^{-35}	7.52×10^{-35}
48	1.93×10^{-33}	9.63×10^{-33}	1.93×10^{-32}
40	4.93×10^{-31}	2.47×10^{-30}	4.93×10^{-30}
32	1.26×10^{-28}	6.31×10^{-28}	1.26×10^{-27}

is very low even with smaller λ values and high network density levels. For example, taking the worst-case scenario (i.e., when high network density and MAC value length $m = 32$ bits are used), the attack complexity, in terms of the average time taken to succeed in an attack with a probability of 50%, is summarised in Table 6.5. It can be concluded that even with the highest truncation level (i.e., when $\lambda = 8$), the time of the attack exceeds the attack timeframe.

Table 6.5: Time to succeed in the random forgery attack in CAS against varied level of hash truncation.

λ **(bits)**	**Time to succeed in the attack with probability 50% (years)**
56	1.04×10^{18}
48	4.08×10^{15}
40	1.59×10^{13}
32	6.22×10^{10}
24	2.43×10^{8}
16	9.50×10^{5}
8	3.71×10^{3}

Based on the above analysis, the weakest link in the security of the CAS hash function is the preimage attack. However, selecting $\lambda_{SEC}^{CAS} = 16$ bits should be sufficient to thwart the risk of this attack.

7. Impact of Hash Value Truncation on Pseudonym Collisions

This section examines the effect of using various levels of hash value truncation on the probability of pseudonym collisions. The intention is to investigate the minimum length of a hash value that will provide adequate pseudonym collision resistance for each of the EAC and CAS schemes based on worst-case scenarios. It is assumed here that there is not a feasible way of filtering out incorrect senders in an event of a pseudonym collision. This assumption allows us to investigate this issue under the worst-case scenario, as if there is another way of getting around the collision issue, then a higher probability of pseudonym collisions can be tolerated and hash value length could go even lower.

Now the question is how to quantify an adequate level of pseudonym collision resistance, or in other words, what is the threshold (i.e., maximum) probability of hash collisions that could give us zero collisions among the pseudonyms during the lifetime of the nodes in the network. When a node sends a message containing a pseudonym, the pseudonym is either checked against a set of pseudonyms stored in the receiving nodes' pseudonym tables as in the case of EAC, or against a list of pseudonyms that are computed by the receiving nodes upon receiving the pseudonym as in the case of CAS. In both cases, there is a set of pseudonyms for which collisions should not occur when a message containing a pseudonym is sent. A collision is said to occur when a pseudonym received in a message matches with two or more entries in a set. These sets change in each data collection period. So, we need to ensure that pseudonym collisions are unlikely in all these sets during the lifetime of a node. To achieve this, we first need to discover how many times these sets change during the lifetime of a node; this can be computed as follows:

The number of pseudonym sets in which collisions should not occur = T_N/T_{DC}, where T_N is the lifetime of a sensor node and T_{DC} is the data collection period.

The threshold probability of hash collisions is then the reciprocal of the resulting number of pseudonym sets, i.e., T_{DC}/T_N. In Section 6, we assumed values for T_N as 50 years and T_{DC} as 1 minute. Based on this, the threshold probability of hash collisions = 1 minute / 50 years = 3.81×10^{-8}. In the rest of this section, we explore the minimum level of hash value truncation that will provide a probability of hash collision that is lower than the specified threshold for EAC and CAS.

7.1. EAC

To investigate the impact of hash value truncation on pseudonym collisions in EAC, we consider two example scenarios that have been illustrated in Figure 7.1 and are described below.

Scenario 1: Node A sends a message to node B. Node B has only one matching record in its pseudonym table and can successfully receive the message. However, another node, C, located in the neighbourhood can also overhear the message and has a matching record in its pseudonym table. So node C will receive the same message despite not being the intended recipient.

Scenario 2: Node A sends a message to node B. Node B has two matching records in its pseudonym table. Node B cannot decide which node is the correct sender of the message.

Assuming that each node has a neighbourhood of b nodes where each node has b records in its pseudonym table, we want to find a minimum λ value, for which there is no collision between a received pseudonym and any of the pseudonyms maintained in the nodes' pseudonym tables (i.e., within the b^2 number of records). Based on the birthday paradox, the probability of collisions can be estimated using Equation (7.1).

$$P(2^{\lambda}, b^2) = 1 - e^{-\left(\frac{b^2 \times (b^2 - 1)}{2 \times 2^{\lambda}}\right)} \tag{7.1}$$

Table 7.1 shows the probability of pseudonym collisions under different network density and λ value settings. From the table, it can be seen that, the collision probability is within the threshold limit when λ = 56 bits (with the

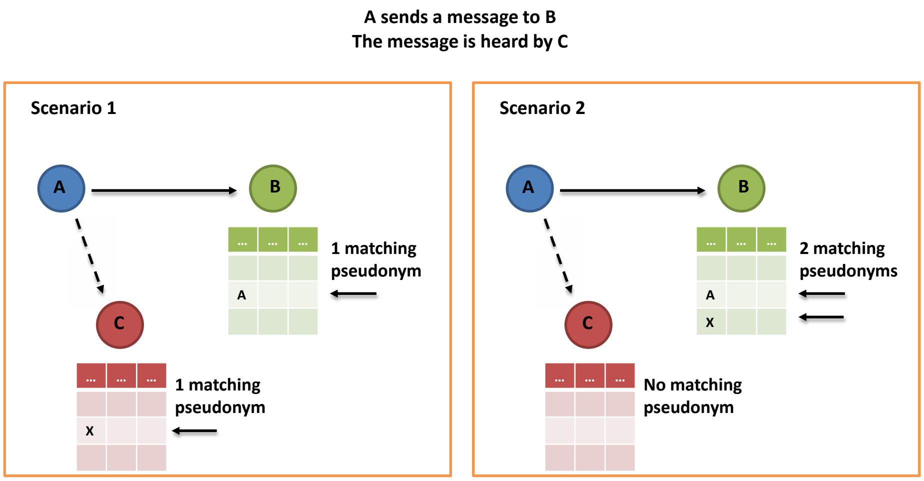


Figure 7.1: Pseudonym collision scenarios in EAC.

network density level set to high), $\lambda = 48$ bits (with the network density level set to medium) and $\lambda = 40$ bits (with the network density level set to low). Therefore, we choose $\lambda_{CR}^{EAC} = 56$, 48 and 40 bits as a recommended collision resistance truncation level for high, medium and low network density levels respectively.

7.2. CAS

In CAS, when a message arrives, a receiving node performs three verifications, and, if and only if all three are positive, the receiver accepts the pseudonym contained in the message. The first verification checks whether the received index value matches with one contained in the receiver's pseudonym table; the second verification checks whether the received message sequence number is larger than the one stored in the receiver's pseudonym table; and the third verification computes a hash value using the KHF and compares it with the received hash value.

Let us use a scenario to illustrate the possible pseudonym collisions in CAS. As illustrated in Figure 7.2, node A has a neighbourhood of three nodes, and node A sends a message to node B. The two other nodes in the neighbourhood, C and D, overhear the transmitted message. All three nodes, B, C and D, will

Table 7.1: Probability of collisions vs. λ values and network density levels in EAC.

λ (bits)	**Probability of collisions**		
	Low density (b=10)	**Medium density (b=50)**	**High Density (b=100)**
120	3.72×10^{-33}	2.35×10^{-30}	3.76×10^{-29}
112	9.53×10^{-31}	6.02×10^{-28}	9.63×10^{-27}
104	2.44×10^{-28}	1.54×10^{-25}	2.47×10^{-24}
96	6.25×10^{-26}	3.94×10^{-23}	6.31×10^{-22}
88	1.60×10^{-23}	1.01×10^{-20}	1.62×10^{-19}
80	4.10×10^{-21}	2.58×10^{-18}	4.14×10^{-17}
72	1.05×10^{-18}	6.62×10^{-16}	1.06×10^{-14}
64	2.68×10^{-16}	1.70×10^{-13}	2.71×10^{-12}
56	6.87×10^{-14}	4.34×10^{-11}	6.94×10^{-10}
48	1.76×10^{-11}	1.11×10^{-8}	1.78×10^{-7}
40	4.50×10^{-9}	2.84×10^{-6}	4.55×10^{-5}
32	1.13×10^{-6}	7.27×10^{-4}	1.20×10^{-2}
24	2.95×10^{-4}	1.70×10^{-1}	9.49×10^{-1}
16	7.30×10^{-2}	1	1
8	1	1	1

verify the index and the sequence number contained in the message. Node B will find a single record in its pseudonym table as it is the intended recipient. Node C will be unable to find a matching index value in its pseudonym table as it is not the intended recipient, so will discard the message. However, if we assume node D could find a matching index in its pseudonym table, and if the verification of the message sequence number is also positive, node D will compute a hash value and compare it with the received hash value. A collision occurs if this hash verification is also positive. In this scenario, only 66% of A's neighbourhood nodes actually evaluate (compute and compare) the hash value. To generalize from this pseudonym collision scenario, let z be the percentage of a neighbourhood of b nodes that have positively completed the first two verifications, and also assume that each of the $(z \times b)$ nodes will only need to evaluate the KHF for a single record in its pseudonym table. The probability of collisions given λ as the truncated hash value length is given in Equation (7.2).

$$P(2^{\lambda}, (z \times b)) = 1 - e^{-\left(\frac{(z\times b)\times((z\times b)-1)}{2\times 2^{\lambda}}\right)} \tag{7.2}$$

For different λ sizes, Table 7.2 shows the probability of collisions under three network density value settings and $z =$ 10%, 50% and 100%, respectively.

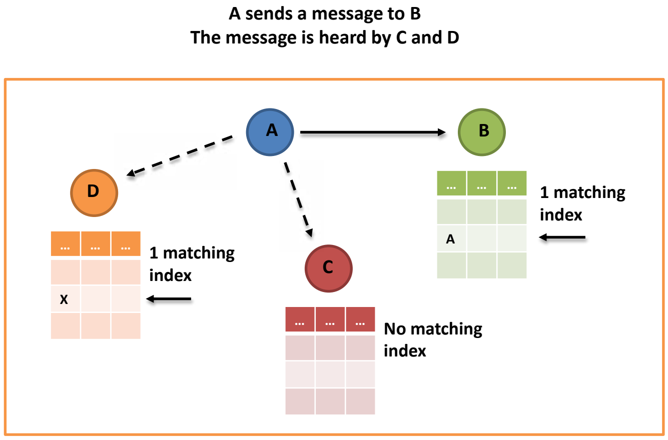


Figure 7.2: Pseudonym collision scenario in CAS.

From the table above, we can see that selecting $\lambda = 40$ bits provides collision probabilities that are lower than the specified threshold value at the worst-case scenario where z=100% and the network density is set to high and medium levels, and when the network density level is set to low, the safe length of the truncated hash value is $\lambda = 32$. Therefore, we recommend $\lambda_{CR}^{CAS} = 40$ bits as the collision resistance truncation level for high and medium density networks and $\lambda_{CR}^{CAS} = 32$ for low density networks.

8. Impact of Hash Value Truncation on Energy Consumption

In this section, we assume that the ID anonymity scheme has a facility to resolve pseudonym collisions, for example, this may be done by performing additional evaluation of a received message such as evaluating the message digest

Table 7.2: Probability of collisions vs. λ, z and network density values in CAS.

z	λ (bits)	**Probability of collisions**		
		Low density (b=10)	**Medium density (b=50)**	**High Density (b=100)**
10% of the nodes evaluate the KHF	56	0	1.00×10^{-16}	7.00×10^{-16}
	48	0	3.55×10^{-14}	1.60×10^{-13}
	40	0	9.10×10^{-12}	4.09×10^{-11}
	32	0	2.33×10^{-9}	1.05×10^{-8}
	24	0	5.96×10^{-7}	2.68×10^{-6}
	16	0	1.53×10^{-4}	6.86×10^{-4}
	8	0	3.80×10^{-2}	1.61×10^{-1}
50% of the nodes evaluate the KHF	56	1.00×10^{-16}	4.22×10^{-15}	1.70×10^{-14}
	48	3.55×10^{-14}	1.07×10^{-12}	4.35×10^{-12}
	40	9.10×10^{-12}	2.73×10^{-10}	1.11×10^{-9}
	32	2.33×10^{-9}	6.99×10^{-8}	2.85×10^{-7}
	24	5.96×10^{-7}	1.79×10^{-5}	7.30×10^{-5}
	16	1.53×10^{-4}	4.57×10^{-3}	1.90×10^{-2}
	8	3.80×10^{-2}	6.90×10^{-1}	9.92×10^{-1}
100% of the nodes evaluate the KHF	56	7.00×10^{-16}	1.70×10^{-14}	6.87×10^{-14}
	48	1.60×10^{-13}	4.35×10^{-12}	1.76×10^{-11}
	40	4.09×10^{-11}	1.11×10^{-9}	4.50×10^{-9}
	32	1.05×10^{-8}	2.85×10^{-7}	1.15×10^{-6}
	24	2.68×10^{-6}	7.30×10^{-5}	2.95×10^{-4}
	16	6.86×10^{-4}	1.90×10^{-2}	7.30×10^{-2}
	8	1.61×10^{-1}	9.92×10^{-1}	1

or MAC by the receiving node. The additional evaluations will result in extra computational cost imposed on the receiving node. In other words, if we use hash value truncation, which means that shorter pseudonyms will be used, then the energy cost on transmitting messages containing the pseudonyms by the sending node will be reduced. However, as shorter pseudonyms will lead to more pseudonym collisions, and more collisions means the receiving node would do more computation to resolve the collided pseudonyms, hence the energy cost on receiving the messages by the receiving node will be increased. This means that the energy cost at the receiving node may offset the energy saved by the sending node. This section analyses the trade-off in energy consumption at the sending and receiving nodes when hash value truncation is used. Our intention in this section is to find an optimal truncated hash value to minimize the total energy costs incurred by both the sender and the receiver.

The analysis is based on the energy model specified in [39] for the TelosB sensor. Tables 8.1 gives the energy costs of the common operations of the sensor.

Table 8.1: Energy costs of common operations of the TelosB sensor running at 4MHz for a measured data rate of 94 kbps (claimed rate is 250 kbps) [39].

Operation	**Energy cost**
Compute for 1 clock cycle (E_C)	1.20 nJ
Transmit 1 bit (E_T)	0.72 μJ
Receive 1 bit (E_R)	0.81 μJ
Listen for 1 clock cycle (E_L)	15.00 nJ
Sleep for 1 clock cycle (E_S)	9.00 pJ

We first estimate the amount of energy gained by transmitting a shorter message based on a single message forwarded by an intermediate node to its one hop neighbours. This result, estimated based on a single message transmission over a single hop, can then be applied to any number of message transmissions and over any number of hops. Let γ be the number of bits that have been saved from transmission in a single message over a single hop (i.e. this is the number of bits that are not transmitted in a single message as the result of using hash value truncation), b the number of one-hop neighbouring nodes of the sender, and E_g the predicted energy gain due to the reduction of γ bits in a transmitted message. E_g is calculated using Equation (8.1). For the sake of simplicity, the costs for listening and sleeping have been neglected in Equation (8.1).

$$E_g = (E_T + b \times E_R) \times \gamma \tag{8.1}$$

where E_T and E_R are the energy costs on transmitting and receiving a single bit of data, respectively. The predicted energy loss (E_L) on resolving pseudonym collisions is computed using Equation (8.2), which is based on the probability of collisions (PC) in the total pseudonym space (PS) at a receiving node and the energy consumed by the node in verifying each collided pseudonym in the record (E_v).

$$E_L = PC \times PS \times E_v \tag{8.2}$$

In the following, we estimate these energy gains and losses for the EAC and CAS schemes, respectively.

8.1. EAC

With EAC, let us consider a scenario in which a message generated by a source node i is forwarded by node j to node r en-route to the base station. The format of the message forwarded from node j to node r is:

$M_{j\to r} = AAI_i||OHAI_{j\leftrightarrow r}||E_{k_{j\leftrightarrow r}}\big(AAI_j||AI_i||E_{k_i}(D)||H\big(AAI_j||AI_i||E_{k_i}(D)\big)\big)$

Four pseudonyms ($AAI_i, OHAI_{j\leftrightarrow r}, AAI_j, AI_i$) are carried in this message, and the length of each pseudonym is 16 bytes (128 bits). Let us assume that the length of $E_{k_i}(D)$ and $H\big(AAI_j||AI_i||E_{k_i}(D)\big)$ is also 16 bytes each. Therefore, reducing one byte from each pseudonym in this message would save 32 bits in total, i.e. $\gamma = 32$ bits. The neighbours of node j will receive the message and may find colliding pseudonyms in their pseudonym tables. To identify the right record using a colliding pseudonym, each receiving node would need to do more computation, i.e. decrypt the payload, compute the message digest and compare the computed digest with the received one. Assuming that the cost involved in comparing the digest values is negligible, the energy required to identify a single record using a colliding pseudonym can be estimated based on the execution time benchmarks presented in Table 4.2, i.e.:

$E_{v(EAC)} = (176.38 + 4.64)ms \times 4MHz \times 1.2nj = 868.90\mu J$

Using the collision probability values shown in Table 7.1, for a given length of the hash value used (i.e. a pseudonym length) (λ) we have computed the number of bits saved per EAC message as the result of using the chosen pseudonym length (versus the default length of 128 bits) (γ), the energy gain as the result of transmitting an EAC message containing the pseudonyms with the chosen length (E_g), the energy loss on resolving any pseudonym collisions (E_L), and the difference between the gain and the loss ($E_g - E_L$), and the computed results are presented in Table 8.2. In the last column (showing the ($E_g - E_L$) values), a positive value indicates that the energy gain is greater than the loss and a negative value indicates the energy gain is less than the loss. Negative values are written inside brackets. It can be seen that the maximum energy gain is obtained when $\lambda = 40$ bits (with the network density level set to high and medium) and $\lambda = 24$ bits (with the network density level set to low). Further reduction of the λ value would result in lower energy gain, or even worse, energy loss on resolving collisions would exceed energy gain through transmission of shorter pseudonyms. Therefore, for the EAC scheme and on the consideration of energy trade-off, we recommend a pseudonym length (i.e., λ_E^{EAC}) of 40 bits when the network density level is medium to high and 24 bits when the network density level is low.

Table 8.2: Energy cost analysis against pseudonym lengths in EAC.

Network density	λ	γ	E_g (μJ)	E_L (μJ)	**Energy gain/(loss)** $E_g - E_L$ (μJ)
Low	120	32	282.24	3.24×10^{-28}	282.24
	112	64	564.48	8.28×10^{-26}	564.48
	104	96	846.72	2.12×10^{-23}	846.72
	96	128	1128.96	5.43×10^{-21}	1128.96
	88	160	1411.20	1.39×10^{-18}	1411.20
	80	192	1693.44	3.56×10^{-16}	1693.44
	72	224	1975.68	9.11×10^{-14}	1975.68
	64	256	2257.92	2.33×10^{-11}	2257.92
	56	288	2540.16	5.97×10^{-09}	2540.16
	48	320	2822.40	1.53×10^{-06}	2822.40
	40	352	3104.64	3.91×10^{-04}	3104.64
	32	384	3386.88	9.84×10^{-02}	3386.78
	24	416	3669.12	25.63	3643.49
	16	448	3951.36	6342.94	(2391.58)
	8	480	4233.60	86889.60	(82656.00)
Medium	120	32	1319.04	5.10×10^{-24}	1319.04

	112	64	2638.08	1.31×10^{-21}	2638.08
	104	96	3957.12	3.35×10^{-19}	3957.12
	96	128	5276.16	8.57×10^{-17}	5276.16
	88	160	6595.20	2.19×10^{-14}	6595.20
	80	192	7914.24	5.61×10^{-12}	7914.24
	72	224	9233.28	1.44×10^{-09}	9233.28
	64	256	10552.32	3.68×10^{-07}	10552.32
	56	288	11871.36	9.42×10^{-05}	11871.36
	48	320	13190.40	2.41×10^{-02}	13190.38
	40	352	14509.44	6.17	14503.27
	32	384	15828.48	1579.22	14249.26
	24	416	17147.52	369280.80	(352133.28)
	16	448	18466.56	2172240.00	(2153773.44)
	8	480	19785.60	2172240.00	(2152454.40)
High	120	32	2615.04	3.27×10^{-22}	2615.04
	112	64	5230.08	8.37×10^{-20}	5230.08
	104	96	7845.12	2.14×10^{-17}	7845.12
	96	128	10460.16	5.48×10^{-15}	10460.16
	88	160	13075.20	1.40×10^{-12}	13075.20
	80	192	15690.24	3.59×10^{-10}	15690.24
	72	224	18305.28	9.17×10^{-08}	18305.28
	64	256	20920.32	2.35×10^{-05}	20920.32
	56	288	23535.36	6.03×10^{-03}	23535.35
	48	320	26150.40	1.54	26148.86
	40	352	28765.44	395.09	28370.35
	32	384	31380.48	104267.52	(72887.04)
	24	416	33995.52	8245823.04	(8211827.52)
	16	448	36610.56	8688960.00	(8652349.44)
	8	480	39225.60	8688960.00	(8649734.40)

8.2. CAS

Each CAS message, $M_{uv} = SID||RID||EncryptedPayload||seq_{uv}||MAC(EncryptedPayload||seq_{uv})$, carries 2 pseudonyms ($SID$,$RID$), so reducing one byte from each pseudonym gives $\gamma = 16$ bits. We assume the worst-case scenario where a received pseudonym is further verified by all receiving nodes (i.e., $z = 100\%$). To resolve each collision, a receiving node needs to compute a MAC value and compare the computed MAC value with the received one. Assuming a payload of 16 bytes and a sequence number of 8 bytes and based on the HMAC-SHA1 execution time shown in Table 4.2, the energy required to verify each collided pseudonym in CAS, $E_{v(CAS)}$, is:

$E_{v(CAS)} = (14.84)ms \times 4MHz \times 1.2nj = 71.23\mu J$

Using the collision probability values shown in Table 7.2 for a given length of the pseudonym hash value used, we have estimated the energy gain (E_g), energy loss (E_L) and the difference between the gain and the loss ($E_g - E_L$) in a similar way as described for EAC (Section 8.1). The results are shown in Table 8.3. It can be concluded, from the table, that the maximum energy gain is obtained when $\lambda = 16$ bits (with the network density level set to high and medium) and $\lambda = 8$ bits (with the network density level set to low). Therefore, we recommend a pseudonym hash value length, $\lambda_E^{CAS} = 16$ bits when the network density level is medium to high and $\lambda_E^{CAS} = 8$ bits when the network density level is low.

9. Impact of Hash Value Truncation on End-to-End Packet Delivery Delays

As discussed earlier, pseudonym collisions at intermediate nodes connecting a source node to the base station may increase the time in delivering a message containing the pseudonyms (i.e. they increase the end-to-end packet delivery delay). The maximum end-to-end (e2e) packet delivery delay should not exceed the data collection period. In this section, we investigate the impact of using hash value truncation in achieving ID anonymity on the e2e packet delivery delay and, based on this investigation, we hope to identify an optimal truncated hash value length for a given application setting, in terms of minimising e2e packet delivery delays.

We use simulation to conduct this investigation. The simulation uses the CASTALIA simulator [40] that is specially designed for WSNs and Body Area Networks, based on the discrete event simulator OMNET++ [41]. The simu-

Table 8.3: Energy trade-off analysis in CAS.

Network density	λ	γ	E_g (μJ)	E_L (μJ)	**Energy gain/(loss)** $E_g - E_L$ (μJ)
Low	56	16	141.12	4.99×10^{-13}	141.12
	48	32	282.24	1.14×10^{-10}	282.24
	40	48	423.36	2.92×10^{-08}	423.36
	32	64	564.48	7.47×10^{-06}	564.48
	24	80	705.60	1.91×10^{-03}	705.60
	16	96	846.72	0.49	846.23
	8	112	987.84	114.68	873.16
Medium	56	16	659.52	6.05×10^{-11}	659.52
	48	32	1319.04	1.55×10^{-08}	1319.04
	40	48	1978.56	3.97×10^{-06}	1978.56
	32	64	2638.08	1.02×10^{-03}	2638.08
	24	80	3297.60	0.26	3297.34
	16	96	3957.12	67.67	3889.45
	8	112	4616.64	3533.11	1083.53
High	56	16	1307.52	4.90×10^{-10}	1307.52
	48	32	2615.04	1.25×10^{-07}	2615.04
	40	48	3922.56	3.21×10^{-05}	3922.56
	32	64	5230.08	8.21×10^{-03}	5230.07
	24	80	6537.60	2.10	6535.50
	16	96	7845.12	519.99	7325.13
	8	112	9152.64	7123.20	2029.44

lated network consists of a large number of nodes (> 500) placed in an area of $200m \times 200m$. Other parameter value settings are summarised in Table 9.1. Using the simulation we investigate the effects of different value settings for the three parameters, i.e. the number of hops, network density and pseudonym length, to study the effects of these parameter values on the e2e packet delivery delays. Both EAC and CAS schemes were implemented at the application level. The MAC and network layers were modified to ensure that all the received packets are passed to the application layer.

Three different network density levels, i.e. low, medium and high, are simulated by adjusting nodes' transmission power levels. The higher a node's transmission power level, the more neighbouring nodes with which this node could connect, thus the higher the network density level. The network density level reflects the size of a pseudonym table maintained at each node. The "Connectivity Map" application in the CASTALIA simulator was used to measure the average pseudonym table size when a node's transmission power is being adjusted. The three network density levels, along with the neighbourhood size and the corresponding power levels, are given below:

1. -10 dBm for low network density ($b \approx 11$ nodes).
2. -3 dBm for medium network density ($b \approx 50$ nodes).
3. 0 dBm for high network density ($b \approx 80$ nodes).

At each network density level, nodes with different numbers of hops away from the base station are assumed. Data collected by a source node are transmitted to the base station in the following order: nodes that are up to 2 hops away from the base station transmit their data first, nodes that are up to 3 hops away transmit their data next, and so on. For each set of transmissions, the maximum e2e packet delivery delays are measured with varying lengths of pseudonyms. The reason for measuring the maximum e2e packet delivery delays, rather than the average e2e packet delivery delays, is that we want to select a pseudonym length that could work efficiently under the worst-case scenario that is bounded by a given data collection period. In other words, in the worst-case scenario (i.e. when suffering the longest delay), data could still be delivered to the base station. To make the results collected from the simulation statistically significant, each simulation is repeated for 30 runs and an average value is calculated from each set of the 30 runs. CASTALIA has several randomised processes, e.g., wireless medium module and MAC module decisions. Additionally, a random startup delay is added to each node (between 0 and 1s).

Cryptographic primitives were implemented by integrating the cryptographic library (CRYPTO++ v.5.6.2) into CASTALIA. However, as CASTALIA does not have a processor module, the execution times of the cryptographic algorithms are simulated by adding corresponding delays. The amount of time delay for each algorithm running on TelosB mote is based on the benchmark shown in Table 4.2. As in the case for some WSN operating systems such as TinyOs [42], we also assume that the operating system does not support multitasking, so if an intermediate node receives more than one packet, and/or has more than one task to process at any given time, the node will suffer correspondingly more delay.

Table 9.1: Simulation parameter values.

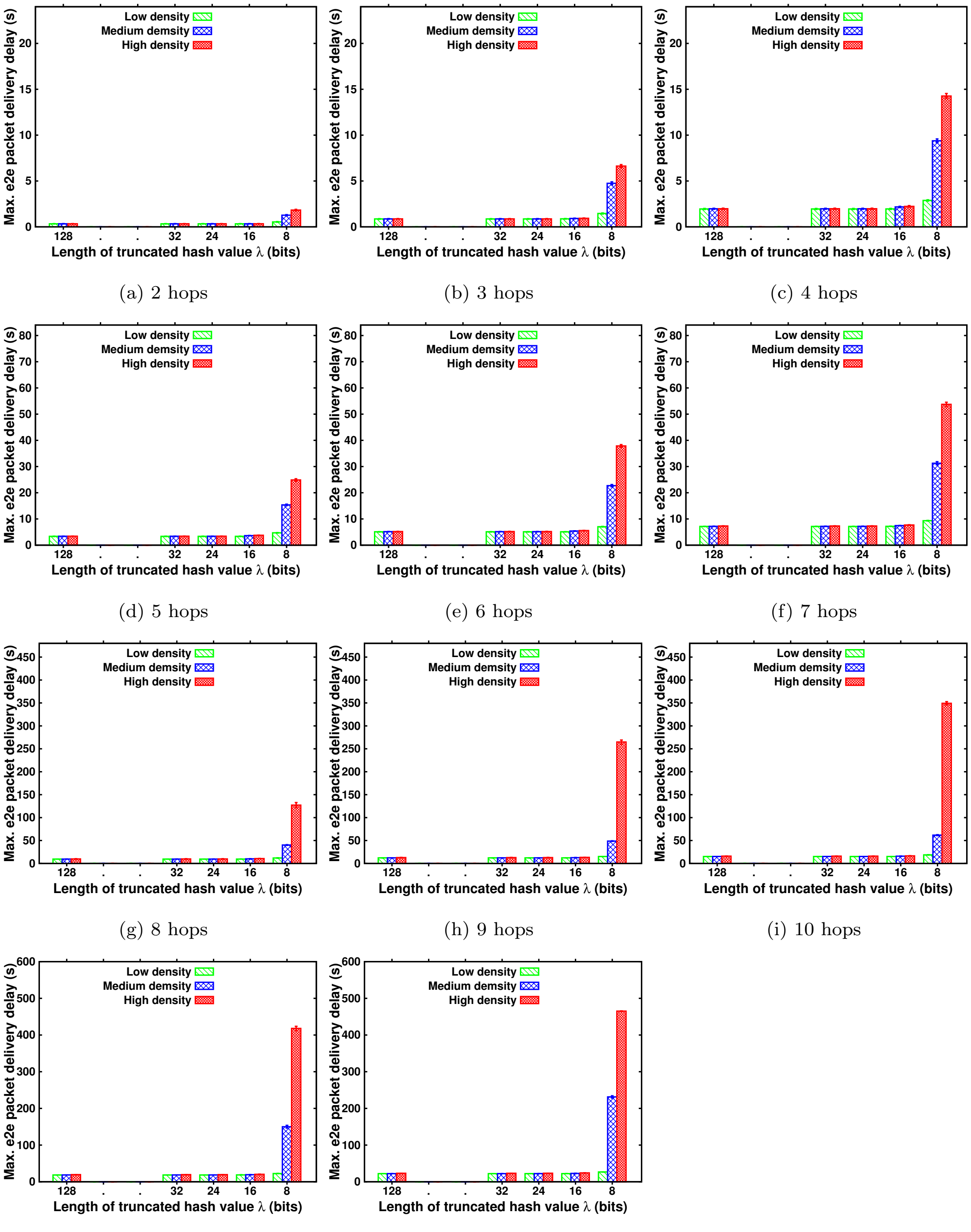
Parameter	**Value**
Simulator version	CASTALIA 3.3
OMNET++	4.6
Number of nodes	576 sensor nodes and 1 base station
Field size	$200m \times 200m$
Sensor nodes distribution	24×24 grid
Base station location	Centre
Simulation time	1800 s
Data packet rate	1 packet/minute
Start-up delay	Between 0 and 1 s
Number of repetitions (runs)	30

In the remaining part of this section, we report our simulation study of the effects of hash value truncation on the maximum e2e packet delivery delays based on the EAC and CAS schemes, respectively.

9.1. EAC

EAC works on a flat topology, so we have set the maximum number of hops to 12. The simulation results are shown in Figure 9.1. It is worth mentioning that, when the higher length hash values are used, e.g., λ =120 to 40 bits, no pseudonym collisions were recorded, and, in such cases, there are no additional delays imposed on intermediate nodes. Therefore, the simulation results recorded for these cases are not reported in the figures below.

From the results shown in Figure 9.1, we can see that, for all the cases of



(a) 2 hops

(b) 3 hops

(c) 4 hops

(d) 5 hops

(e) 6 hops

(f) 7 hops

(g) 8 hops

(h) 9 hops

(i) 10 hops

(j) 11 hops

(k) 12 hops

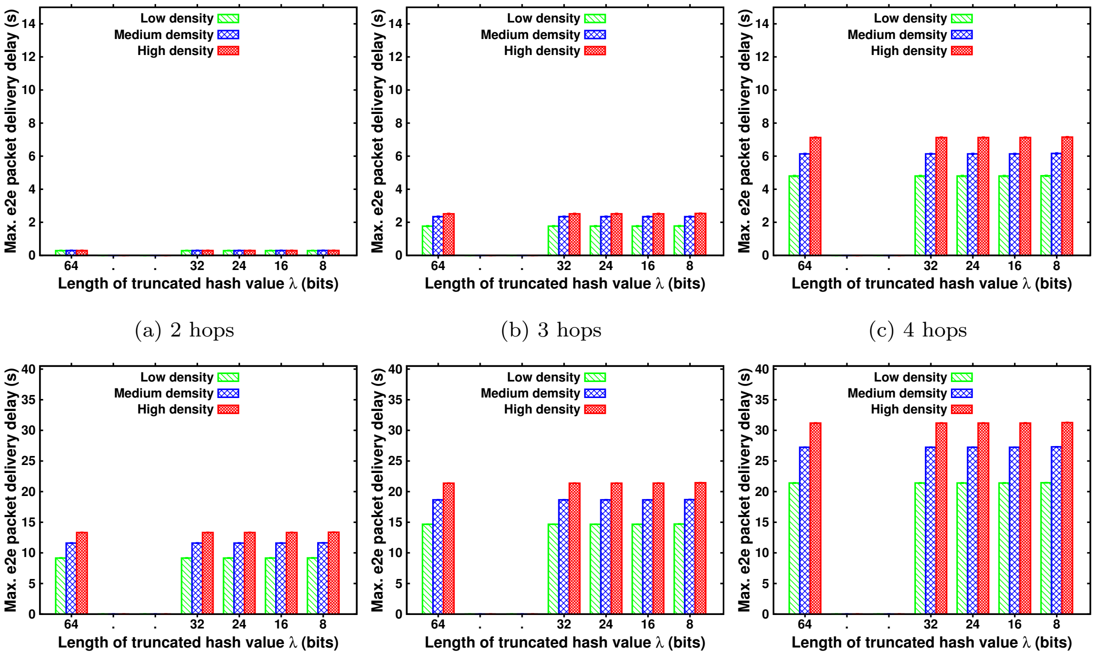
Figure 9.1: EAC maximum e2e packet delivery delays vs. truncated hash values and hop counts.

hop counts, when the pseudonym length, λ, is reduced from 128 bits to 16 bits, the e2e packet delivery delays have very little changes. In other words, when the pseudonym length ≥ 16, the e2e packet delivery delays are not sensitive to any pseudonym length changes, although the higher the hop counts, the higher the e2e packet delivery delays. In addition, the results also show that, when the pseudonym length ≥ 16, the delays are not sensitive to network density levels.

However, when the pseudonym length is further reduced to below 16-bits, i.e., when λ is set to 8 bits, we can see a different set of results. Firstly, there is a marked increase in the e2e packet delivery delays, and the higher the network density level and the higher the hop count, the bigger the increase. These simulation results have further proved our early analysis that, when the pseudonym length is set too low, the number of pseudonym collisions will increase sharply. The intermediate nodes, upon the receipt of the collided pseudonyms, will need to do additional computations to identify the nodes referenced by the collided pseudonyms. The additional computations will add more delays into the e2e packet delivery delay, and, in addition, the additional computations will make the nodes busier causing further delays (queuing delays) in processing incoming messages. This increase in the e2e packet delivery delays will get worse when the network density level gets higher and/or when the hop count gets bigger. This is because when the density gets higher, the pseudonym space in pseudonym tables will increase causing more collisions. When the hop counts gets bigger, more of these effects will be added into the e2e packet delivery delay. It is also worth noting that when the parameter values are set as: $\lambda = 8$, the number of hops is 8 or higher and the network density level is medium or high, the maximum e2e packet delivery delay is more than 1 minute, exceeding the data collection period. When the delay increases beyond the data collection period, the messages containing the data collected in the previous period are not processed and delivered, before the messages containing the data collected in the current period arrive at the intermediate nodes. This causes significant queuing delays at these nodes, and as a result, the e2e packet delivery delays will increase sharply. In other words, the pseudonym length should be chosen such that the resulting e2e packet delivery delays should not be adversely affected, and based on the sole consideration of this single parameter, we can choose a λ value of 16 bits for the EAC scheme, i.e. $\lambda_D^{EAC} = 16$ bits.

9.2. CAS

CAS is designed for a clustered network topology, so typically there is a smaller number of hops between a pair of communication nodes in comparison with EAC. The maximum number of hops for the grid node deployment assumed for CAS in our investigation is set to 7 based on an average cluster size of 4 nodes per cluster. The simulation results are shown in Figure 9.2.



(a) 2 hops

(b) 3 hops

(c) 4 hops

(d) 5 hops

(e) 6 hops

(f) 7 hops

Figure 9.2: CAS maximum e2e packet delivery delay vs. length of truncated hash values and hop counts.

From Figure 9.2, we can make the following observations. Firstly, for all the hop count cases, the CAS performs best when the network density level is the lowest, and worst when the network density level is the highest. The differences in the e2e packet delivery delays when different network density levels are set are more obvious in CAS compared to EAC. This is because the receiving nodes in CAS need to always compute the hash value upon receiving a message causing more delays when the number of messages increases in dense networks, whereas in EAC the receiving nodes only need to find a matching pseudonym in their pseudonym table and the computation of the next pseudonym (i.e., hash value) is done after confirming that there is a pseudonym match.

Secondly, the bigger the hop count, the higher the maximum e2e packet delivery delay. This result is intuitive, as more hops a message transits, more hash value computations will be involved, thus the higher the delay.

Thirdly, for all the hop count cases, the maximum e2e packet delivery delays are not sensitive to the pseudonym length (λ) changes. This is the case even when the truncated hash value decreases to 8 bits. This result is different from the observation made on the EAC scheme. There are three reasons for this. Firstly, the probability of collisions in CAS is lower than that of EAC. This is because for any received pseudonym in CAS, each node may only have one matching index at most in its pseudonym table whereas in EAC each node may have more than one matching pseudonym in its pseudonym table. The received pseudonyms in CAS are filtered immediately by the index verification and only if they pass the index verification, is the hash value verified. Secondly, if a matching hash value is found in CAS, the next verification will be to compute a MAC value and compare it with the received MAC value. In EAC, however, the verification involves decrypting the message using AES, which is much more costly compared to computing HMAC in CAS (refer to Table 4.2). Thirdly, CAS uses a clustered topology, so packets traverse fewer hops before reaching the base station. All of these factors help to reduce the likelihood of the maximum e2e packet delivery delays exceeding the data collection period.

Based on these considerations, the safe level of truncation in terms of e2e packet delivery delays in CAS is 8 bits, i.e. $\lambda_D^{CAS} = 8$ bits, for all network density levels.

10. Further Discussions

In this section, we summarize the results from our above analyses to determine the most appropriate truncated hash value lengths, λ_{REC}, for the two ID anonymity schemes, EAC and CAS. The decisions are made based on the four criteria attributes, i.e. security (in terms of resistance against brute force attacks on hash functions), pseudonym collisions (in terms of reducing the probability of pseudonym collisions), energy trade-off (in terms of balancing transmission costs at the sending node and computational costs at the receiving nodes) and e2e packet delivery delay (in terms of reducing the delay below the data collection period). The results are summarised in Table 10.1 for each of the three network density levels under the considered parameter settings. For a given case, the recommended truncated hash value size. i.e. λ_{REC}, is the longest one as determined by the involved four attributes depending on whether collision resistance is required or not (refer to Equation (5.1)). From the table, it can be seen that the pseudonym collisions attribute imposes the most constraints on the reduction of the hash value lengths whereas the e2e packet delivery delay imposes the least constraints. The analysis emphasizes the role of the following factors in the determination of an appropriate truncated hash value length:

1. Type of hash function and how it is used in an ID anonymity scheme: The type of hash function, i.e., UHF or KHF, determines the type of brute force attacks an adversary may attempt. In addition, the length and lifetime of the cryptographic key in KHF has an impact on the success of the exhaustive key search attack and consequently on the selection of the optimal hash value length based on security considerations. Further, how these functions are used in an ID anonymity scheme also impacts the hash value truncation level. For example, the CAS scheme uses a sequence number to protect the IDs against the replay attack whereas EAC does not, leaving EAC prone to the replay attack. The pseudonym in the CAS scheme is also designed by combining an index value and a hash value. Such design has an impact on reducing the pseudonym collision

probability, thus enabling the CAS scheme to afford shorter hash values compared to EAC.

2. Sensor node capabilities: The computational capability of the microcontroller and the sensor node energy supply also have an impact on determining the optimal length of the truncated hash value in all four attributes. For example, the more time it takes for a sensor node to compute hash values to address the pseudonym collision problem, the more energy the node will consume and at the same time the higher the e2e packet delivery delay will be introduced. In addition, the energy supply of a sensor node determines the lifetime of the node and consequently affects the time available for an adversary to successfully launch an attack using some of the brute force attack methods, e.g., the preimage attack in CAS. It also impacts the pseudonym collision probability threshold. The shorter the lifetime of a node, the higher the pseudonym collision probability threshold we can use, and accordingly the shorter the hash value (i.e. the more hash value truncation) we can afford to apply.
3. Adversary capabilities: The adversary capabilities determine the speed at which an adversary can perform the offline and online brute force attacks. For example, offline attacks are affected by the speed of computing hash values using brute force. Online attacks are affected by the transmission capabilities of the adversary (should the sensor nodes transceiver allow reception at the same rate). Any changes made in these computational and communication capabilities of the adversary will have an impact on the recommended level of truncation imposed on the hash value based on the security considerations.
4. Ability to resolve pseudonym collisions: If an ID anonymity scheme that is designed based on a hash function do not have a built-in facility to identify a correct sender should a pseudonym collision occur, then the scheme can only tolerate up to a certain threshold level of a pseudonym collision probability. This imposes constraints on the level of hash value truncation that could be applied. However, if the collision resistance is not an over-

riding requirement, then the threshold level of the pseudonym collision probability may be increased and therefore a shorter hash value may be chosen provided the constraints imposed by the other two attributes, i.e., energy trade-off and e2e packet delivery delay, allow. For example, in the case of using EAC in a low density network, the recommended truncated hash value length is 40 bits when the collision resistance property becomes a requirement. However, the hash value length can be reduced to 24 bits if the collision resistance property is not required

5. Network density: The network density implies the size of a node neighbourhood or the number of single-hop neighbours a node has. This number also determines the number of records each node should maintain in its pseudonym table. The network density plays a major role in deciding on the hash value truncation level in all the four attributes. For example, in terms of security considerations, the lower the network density level, the lower the chance for an adversary to succeed using the random forgery attack for both UHF and KHF. In terms of pseudonym collision considerations, reducing the network density levels reduces the pseudonym collision probability and consequently more truncation can be applied. In terms of energy trade-off, the lower the network density level, the lower the pseudonym collisions probability, and, as a result, less computations will be required to resolve any collisions, thus the energy gains may be maximized at a higher level of hash value truncation. Similarly, in terms of e2e packet delivery delay, reducing the network density level reduces the pseudonym collision probability which will result in less time being used to address the collisions and consequently less e2e packet delivery delay.
6. Data collection rate: The data collection rate plays an important role in the estimation of the optimal truncation level for the security and pseudonym collision attributes. If the data collection rate is reduced, the data collection period will increase and this will give an adversary more time to launch a successful brute-force attack (e.g., the replay attack in EAC). As a result, a lower level of hash value truncation (i.e., a longer

hash value length) should be used. Similarly, if the data collection period is increased, the threshold probability of hash collisions will increase, which will allow for a greater hash value truncation (i.e., a shorter hash value length). Obviously, the data collection rate affects the hash value truncation level in two ways. One is affected by the security attribute, the other is by the pseudonym collision attribute, and there is a trade-off between the two.

Table 10.1: Summary of findings.

Criteria	**Collision resistance is required**		**Collision resistance is not required**	
	EAC	**CAS**	**EAC**	**CAS**
Low density				
Security λ_{SEC}	24	16	24	16
Pseudonym collisions λ_{CR}	40	32	-	-
Energy trade-off λ_E	-	-	24	8
End-to-end packet delivery delay λ_D	-	-	16	8
Recommended λ_{REC}	40	32	24	16
Medium density				
Security λ_{SEC}	24	16	24	16
Pseudonym collisions λ_{CR}	48	40	-	-
Energy trade-off λ_E	-	-	40	16
End-to-end packet delivery delay λ_D	-	-	16	8
Recommended λ_{REC}	48	40	40	16
High density				
Security λ_{SEC}	24	16	24	16
Pseudonym collisions λ_{CR}	56	40	-	-
Energy trade-off λ_E	-	-	40	16
End-to-end packet delivery delay λ_D	-	-	16	8
Recommended λ_{REC}	56	40	40	16

From the above analyses using the methodology we have proposed, one can see that, based on the consideration of the four attributes, it is possible to determine an optimal hash value truncation level. The analysis results show that the hash value used in pseudonyms can be truncated significantly. This reduction in the length of a hash value reduces message lengths, which, in turn, will lead to reduced transmission load and bandwidth requirements imposed on sensor nodes. For example, in EAC, each message transmitted from a source node to the base station multi-hop away contains four pseudonyms: the global anonymous identity, the one-hop anonymous identity and two anonymous acknowledgement identities. Similarly, in CAS, when a message is sent from a source node to the base station through other intermediate nodes, the source node will need to construct two pseudonyms: an end-to-end pseudonym and a next-hop pseudonym. Reducing the length of each pseudonym by 1 byte reduces 4 bytes off EAC packet and 2 bytes off CAS packet. The reduction in packet size will lead to less transmission errors, lower frequency of retransmissions and better utilization of the bandwidth [43]. The reduction in the transmission load will also lead to the reduction of energy consumption as discussed in Section 8. These reductions cannot only improve the efficiency of an ID anonymity scheme constructed based on a hash function, but also help to prolong the lifetime of sensor nodes that are typically resource constrained. It should also be emphasised that reducing the pseudonym length can also reduce the memory requirement (i.e. memory overhead) of a sensor node. This is especially the case for EAC where the pseudonyms are stored in the pseudonym table of each node. Figure 10.1 shows the impact on the memory requirements under different network density levels in EAC where the length of the pseudonym used is 40 bits. For a large network density (e.g., $b = 100$ nodes), the memory overhead can be reduced by 29%.

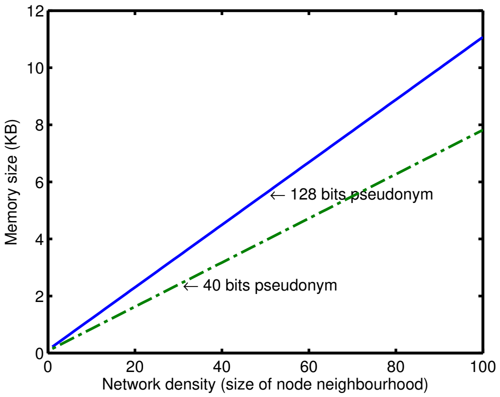


Figure 10.1: Memory requirements vs. pseudonym length and network density in EAC.

11. Conclusions

This paper has presented a comprehensive study of the impacts of hash value truncation on the security and efficiency of ID anonymity schemes designed by using hash functions. The study is carried out based on two existing ID anonymity schemes, EAC and CAS, specifically designed for WSNs. The results from our study have led to some interesting discoveries. Firstly, there are multiple attributes that affect the choice of an optimal length of a truncated hash value. In our analysis, we have identified four such attributes, and these are security (in terms of resistance against brute force attacks on hash functions), pseudonym collisions (in terms of reducing the probability of pseudonym collisions), energy trade-off (in terms of balancing transmission costs at the sending node and computational costs at the receiving nodes) and e2e packet delivery delay (in terms of reducing this delay below the data collection period). Secondly, we have identified six factors that affect the selection of the optimal hash value truncation level based on the above four attributes. These are the type and usage of hash functions, sensor node capabilities, adversary capabilities, ability to resolve pseudonym collisions, network density and data collection rate. Thirdly, the effects of different factors may be contradictory, therefore there are some trade-offs, which should be considered when determining the length of a truncated hash value. The work presented in this paper has demonstrated the use of a systematic analysis methodology for analyzing the effects of hash value lengths in terms of security and performance. Though the work is carried out based on ID anonymity schemes designed for WSNs, the methodology can also be applied to hash function based schemes designed for other problem contexts, such as preserving patients' identity privacy in an e-health cloud. Future work will investigate the impact of sensor node compromise attack on the security of ID anonymity in WSNs.

References

[1] A. Becher, Z. Benenson, M. Dornseif, Tampering with motes: Real-world physical attacks on wireless sensor networks, in: Proceedings of the Third International Conference on Security in Pervasive Computing, SPC'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 104–118. doi:10.1007/11734666_9.

[2] S. Misra, G. Xue, Efficient anonymity schemes for clustered wireless sensor networks, International Journal of Sensor Networks 1 (1-2) (2006) 50–63. doi:10.1504/IJSNET.2006.010834.

[3] Y. Ouyang, Z. Le, Y. Xu, N. Triandopoulos, S. Zhang, J. Ford, F. Makedon, Providing anonymity in wireless sensor networks, in: IEEE International Conference on Pervasive Services, 2007, pp. 145–148. doi:10.1109/PERSER.2007.4283904.

[4] J.-P. Sheu, J.-R. Jiang, C. Tu, Anonymous path routing in wireless sensor networks, in: IEEE International Conference on Communications, 2008. ICC '08., 2008, pp. 2728–2734. doi:10.1109/ICC.2008.515.

[5] J. Chen, X. Du, B. Fang, An efficient anonymous communication protocol for wireless sensor networks, Wireless Communications and Mobile Computing 12 (14) (2012) 1302–1312. doi:10.1002/wcm.1205.

[6] X. Luo, X. Ji, M.-S. Park, Location privacy against traffic analysis attacks in wireless sensor networks, in: International Conference on Information

Science and Applications (ICISA), 2010, 2010, pp. 1–6. doi:10.1109/ICISA.2010.5480564.

[7] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Computer networks 52 (12) (2008) 2292–2330. doi:10.1016/j.comnet.2008.04.002.

[8] PUB-FIPS 180-4, Secure Hash Standard,National Institute of Standards and Technology. , http://csrc.nist.gov/publications/PubsFIPS.html, (2015 - accessed January 13, 2016). doi:10.6028/NIST.FIPS.180-4.

[9] PUB-FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC), National Institute of Standards and Technology. , http://csrc.nist.gov/publications/PubsFIPS.html, (2008 - accessed January 13, 2016).

[10] ISO/IEC 9797-1:2011, Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher. , http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=50375, (2011 - accessed January 13, 2016).

[11] ISO/IEC 9797-2:2011, Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function. , http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=51618, (2011 - accessed January 13, 2016).

[12] H. Krawczyk, M. Bellare, R. Canetti, RFC 2104: HMAC: Keyed-hashing for message authentication. , https://www.ietf.org/rfc/rfc2104.txt, (1997 - accessed January 13, 2016).

[13] M. J. Dworkin, SP 800-38B Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. , NIST Special Publica-

tions 800. , http://csrc.nist.gov/publications/PubsSPs.html, (2005 - accessed January 13, 2016).

[14] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, Handbook of applied cryptography, CRC press, 1996.

[15] B. Preneel, Analysis and design of cryptographic hash functions, Ph.D. thesis, Katholieke Universiteit Leuven (1993).

[16] P. S. Gauravaram, Cryptographic hash functions: cryptanalysis,design and applications, Ph.D. thesis, Queensland University of Technology (2007).

[17] R. Sobti, G. Geetha, Cryptographic hash functions: a review, IJCSI International Journal of Computer Science Issues 9 (2) (2012) 461479.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2009.

[19] S. William, W. Stallings, Cryptography and Network Security,4/E, Pearson Education India, 2006.

[20] K. Jia, X. Wang, Z. Yuan, G. Xu, Distinguishing and second-preimage attacks on cbc-like macs, in: J. Garay, A. Miyaji, A. Otsuka (Eds.), Cryptology and Network Security, Vol. 5888 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 349–361. doi:10.1007/978-3-642-10433-6_23.

[21] B. Preneel, V. Rijmen, State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography, Leuven, Belgium, June 3-6,1997 Revised Lectures, Vol. 1528, Springer Science & Business Media, 1998.

[22] Q. H. Dang, SP 800-107 Recommendation for Applications Using Approved Hash Algorithms. , NIST Special Publications 800. , http://csrc.nist.gov/publications/PubsSPs.html, (2012 - accessed January 13, 2016).

[23] A. A. Nezhad, A. Miri, D. Makrakis, Location privacy and anonymity preserving routing for wireless sensor networks, Computer Networks 52 (18) (2008) 3433–3452. doi:10.1016/j.comnet.2008.09.005.

[24] J.-H. Park, Y.-H. Jung, K.-H. Lee, H. Ko, M.-S. Jun, A new privacy scheme for providing anonymity technique on sensor network, in: International Conference on Ubiquitous Computing and Multimedia Applications (UCMA), 2011, 2011, pp. 10–14. doi:10.1109/UCMA.2011.11.

[25] M. Abdullahi, G. Wang, A lightweight anonymous on-demand routing scheme in wireless sensor networks, in: IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2012, 2012, pp. 978–985. doi:10.1109/TrustCom.2012.36.

[26] J. Deng, Z. Zhang, Orthogonal code based anonymity communication protocol for wireless sensor networks, Sensors & Transducers 172 (6) (2014) 61–66.

[27] F. Xue, P. R. Kumar, The number of neighbors needed for connectivity of wireless networks, Wireless networks 10 (2) (2004) 169–181. doi:10.1023/B:WINE.0000013081.09837.c0.

[28] S. Bandyopadhyay, E. J. Coyle, Minimizing communication costs in hierarchically-clustered networks of wireless sensors, Computer Networks 44 (1) (2004) 1–16. doi:10.1016/S1389-1286(03)00320-7.

[29] Crossbow, Telosb datasheet, http://www.memsic.com/userfiles/files/Datasheets/WSN/TelosB_Datasheet.pdf, (accessed January 13, 2016).

[30] T. Xie, F. Liu, D. Feng, Fast collision attack on md5, IACR Cryptology ePrint Archive, Report 2013/170, http://eprint.iacr.org/ (2013).

[31] E. Barker, A. Roginsky, SP 800-131A Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. , NIST Special Publications 800. , http://csrc.nist.gov/publications/

PubsSPs.html, (2015 - accessed January 13, 2016). doi:10.6028/NIST.SP.800-131Ar1.

[32] W. Backes, J. Cordasco, Moteaodv – an aodv implementation for tinyos 2.0, in: Proceedings of the 4th IFIP WG 11.2 International Conference on Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices, WISTP'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 154–169. doi:10.1007/978-3-642-12368-9_11.

[33] J. Lee, K. Kapitanova, S. H. Son, The price of security in wireless sensor networks, Computer Networks 54 (17) (2010) 2967–2978. doi:10.1016/j.comnet.2010.05.011.

[34] D. Jinwala, K. Dasgupta, D. Patel, Flexisec: A configurable link layer security architecture for wireless sensor networks, Journal of Information Assurance and Security 4 (arXiv: 1203.4697) (2012) 582603.

[35] J. Guo, S. Ling, C. Rechberger, H. Wang, Advanced meet-in-the-middle preimage attacks: First results on full tiger, and improved results on md4 and sha-2, in: M. Abe (Ed.), Advances in Cryptology - ASIACRYPT 2010, Vol. 6477 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 56–75. doi:10.1007/978-3-642-17373-8_4.

[36] M. Dohler, T. Watteyne, T. Winter, D. Barthel, Urban wsns routing requirements in low power and lossy networks, IETF draft,IETF ROLL workgroup.

[37] A. Somov, C. C. Ho, R. Passerone, J. W. Evans, P. K. Wright, Towards extending sensor node lifetime with printed supercapacitors, in: Proceedings of the 9th European Conference on Wireless Sensor Networks, EWSN'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 212–227. doi:10.1007/978-3-642-28169-3_14.

[38] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, SP 800-57 Part 1-Rev.3 Recommendation for Key Management: Part 1: General (Revision 3). ,

NIST Special Publications 800. , http://csrc.nist.gov/publications/PubsSPs.html, (2012 - accessed January 13, 2016).

[39] G. de Meulenaer, F. Gosset, O.-X. Standaert, O. Pereira, On the energy cost of communication and cryptography in wireless sensor networks, in: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2008. WIMOB '08., 2008, pp. 580–585. doi:10.1109/WiMob.2008.16.

[40] A. Boulis, et al., Castalia: A simulator for wireless sensor networks and body area networks, NICTA: National ICT Australia.

[41] A. Varga, et al., The OMNeT++ discrete event simulation system, in: Proceedings of the European simulation multiconference (ESM2001), Vol. 9, sn, 2001, p. 65.

[42] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, Tinyos: An operating system for sensor networks, in: W. Weber, J. Rabaey, E. Aarts (Eds.), Ambient Intelligence, Springer Berlin Heidelberg, 2005, pp. 115–148. doi:10.1007/3-540-27139-2_7.

[43] M. Vuran, I. Akyildiz, Cross-layer packet size optimization for wireless terrestrial, underwater, and underground sensor networks, in: The 27th IEEE Conference on Computer Communications. INFOCOM 2008., 2008, pp. 226–230. doi:10.1109/INFOCOM.2008.54.