

# GDB: Group Distance Bounding Protocols

Srdjan Capkun  
ETH Zurich  
capkuns@inf.ethz.ch

Karim El Defrawy  
(corresponding author)  
UC Irvine  
keldefra@ics.uci.edu

Gene Tsudik  
UC Irvine  
gts@ics.uci.edu

## ABSTRACT

Secure distance bounding (DB) protocols allow one entity, the verifier, to securely obtain an upper-bound on the distance to another entity, the prover. Thus far, DB was considered mostly in the context of a single prover and a single verifier. There has been no substantial prior work on secure DB in group settings, where a set of provers interact with a set of verifiers. The need for group distance bounding (GDB) is motivated by many practical scenarios, including: group device pairing, location-based access control and secure distributed localization. GDB is also useful in mission-critical networks and automotive computer systems. This paper addresses, for the first time, GDB protocols by utilizing the new *passive DB* primitive and the novel *mutual multi-party GDB* protocol. We show how they can be used to construct secure and efficient GDB protocols for various settings. We analyze security and performance of our protocols and compare them with existing DB techniques when applied to group settings.

## 1. INTRODUCTION

Wireless networks – especially, sensor and mobile ad-hoc networks, have become increasingly popular. Enabled by pervasive availability of location information, new wireless scenarios have emerged where accurate proximity information is essential to both applications and basic networking functions. Such scenarios require secure, reliable and efficient verification of distances between nodes, in addition to node authentication. Distance Bounding (DB) can address such scenarios by allowing one entity (verifier) to obtain an upper-bound on the distance to another entity (prover) and, optionally, authenticate the latter. DB was introduced by Brands and Chaum [4] as a means of preventing the so-called “mafia fraud” attacks on bank ATMs<sup>1</sup>. In Brands and Chaum’s DB approach, a user’s smart-card (verifier) checks its proximity to the ATM (prover). DB has been recently implemented [19] using commercial off-the-shelf electronics (resulting in 15cm accuracy). It was also suggested and implemented as a means of securely determining node loca-

<sup>1</sup>A “mafia fraud” attack occurs when the attacker identifies itself to the verifier using the identity of a prover, without the latter being aware (i.e., man-in-the-middle attack).

tions in wireless networks [14, 6, 8, 23].

In most prior work, DB was considered in the context of a single prover and a single verifier. Group Distance Bounding (GDB) is the natural extension of the DB concept to group settings with multiple provers and verifiers. Multiple verifiers provide several advantages including: higher attack resilience and improved availability (by avoiding a single point of compromise or failure), in addition to facilitating localization using multilateration. The common goal in applications that require GDB is: *several devices must securely measure distances between themselves or should only operate in the vicinity of each other*.

GDB is motivated by the following emerging wireless applications: group device pairing – a procedure for setting up an initial secure channel among a group of previously unfamiliar wireless devices. There are several scenarios where this is required, e.g., when an ephemeral ad-hoc group of users meet. Each user has a personal wireless device that must establish a secure channel with devices of other users. One concrete example using cell-phones is described in [11]. Another scenario is that of a single user with multiple devices, e.g., in a home area network [5]. A secure mechanism is required to ensure that the group of communicating devices is clustered within a particular area, i.e., each device is within a certain distance from every other device. The mutual multi-party GDB protocol (Section 3.3) achieves this.

Another application that can benefit from GDB is automotive computer systems. Recent research [10] pointed out vulnerability of such systems to attacks through wireless interfaces (demonstrated in [3] using relay attacks). As more components of such systems communicate wirelessly, it becomes critical to ensure that the origin of communication is from within the car to prevent relay attacks. Ensuring that such components only communicate with each other prevents attacks through unauthorized or outside malicious components. The mutual multi-party GDB protocol (Section 3.3) achieves this.

GDB is also useful in critical, e.g., military, MANETs where a key operational requirement is to track locations of, and authenticate, friendly nodes [2]. Critical MANETs generally operate without any infrastructure and in hostile environments where node compromise is quite realistic. GDB

can be used to implement location based-access control and location-based group key management in critical MANETs. Both mutual multi-party GDB (Section 3.3) and passive DB (Section 3.1) can be used in such settings.

In this paper, we show that a straightforward extension of previous single prover single verifier DB to GDB is inefficient and insecure if used for localization without synchronization between verifiers (which was also pointed out in [8]). We explore and propose more efficient and secure GDB approaches. We make the following contributions:

- *Definition of Group Distance Bounding (GDB)*
- *New primitives: Passive DB and Mutual Multi-Party GDB*
- *A set of secure and efficient GDB protocols*
- *Security and performance analysis of proposed protocols*

The rest of the paper is organized as follows: we overview traditional DB protocols, formulate the GDB problem and state our system and adversary models in Section 2. We present details and security analysis of our GDB building block primitives in Section 3. We then show how to use these building blocks to construct GDB protocols in both one-way and mutual GDB settings in Section 4. We analyze performance and security of GDB protocols in Section 5. We discuss related work in Section 6 and conclude with open issues and future work in Section 7.

## 2. BACKGROUND AND PROBLEM STATEMENT

We begin with an overview of DB protocols, followed by the problem statement and system model.

### 2.1 Overview of Distance Bounding (DB)

Figure 1 shows the generic DB protocol operation. The core of any *one-way DB* protocol is the distance measurement phase, whereby the verifier measures round-trip time between sending its challenge and receiving the reply from the prover. Verifier’s challenges are unpredictable to the prover and replies are computed as a function of these challenges. Thus, the prover cannot reply to the verifier before receiving its challenges. The prover, therefore, cannot pretend to be closer to the verifier than it really is (only further). First, the verifier and the prover each generate  $n$   $b$ -bit nonces  $c_i$  and  $r_i$  ( $1 \leq i \leq n$ ), respectively. In the Brands-Chaum DB protocol [4], the prover also commits to its nonces (using any secure commitment scheme). The verifier sends all  $c_i$  to the prover, one at a time. Once each  $c_i$  is received, the prover computes, and responds with a function of its own nonce and that of the verifier,  $f(c_i, r_i)$ . The verifier checks the reply and measures the elapsed time between each challenge and response. The process is repeated  $n$  times and the protocol completes successfully only if *all*  $n$  rounds succeed and all responses correspond to prover’s committed value. The processing time on the prover’s side  $\alpha = t_s^P - t_r^P$  must be negligible (compared to the time of flight); otherwise, a computationally powerful prover could claim a false bound. This time might be tolerably small, depending on the un-

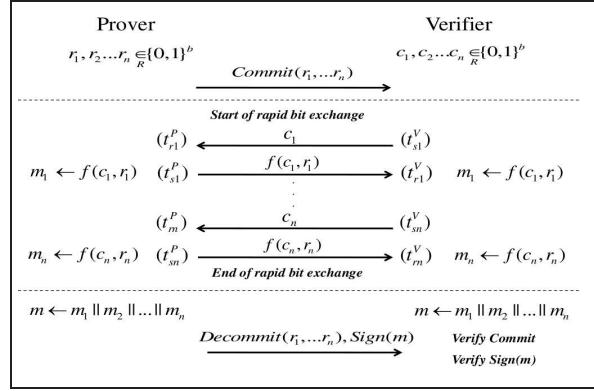


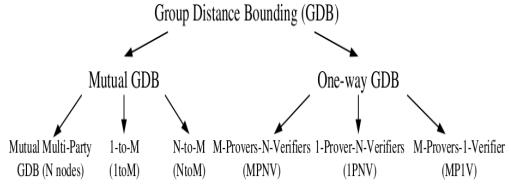
Figure 1: Basic DB Operation.

derlying technology, the distance measured and the required security guarantees (less than 1nsec processing time yields 0.15m accuracy [19]).

Security of DB protocols relies on two assumptions: (1) challenges are random, and unpredictable to the prover before being sent by the verifier, and (2) challenges traverse the distance between the prover and the verifier at maximum possible speed. i.e., the speed of electromagnetic waves. After executing a DB protocol, the verifier knows that distance to the prover is at most  $\frac{t_r^V - t_s^V - \alpha}{2} \cdot c$ , where  $\alpha$  is the processing time of prover (ideally, negligible) and  $c$  the speed of light [4]. DB protocols typically require  $(2n + \mathcal{C})$  messages, where  $\mathcal{C}$  is the number of messages exchanged in the pre- and post-processing protocol phases. Typically,  $\mathcal{C} \ll n$  and thus can be ignored.

In some cases (e.g., distributed localization), there is a need for mutual DB between two parties:  $P_1$  and  $P_2$ . This can be achieved by modifying the one-way DB protocol such that each response from  $P_2$  to a challenge by  $P_1$  also includes a challenge from  $P_2$  to  $P_1$ . This requires  $2n + 2\mathcal{C} + 1$  messages instead of  $2(2n + \mathcal{C})$  for mutual DB and is shown in [25]. Both parties generate and commit to two random bit strings  $[c_1, c_2, \dots, c_n]$  and  $[s_1, s_2, \dots, s_n]$ .  $P_1$  starts by sending the first challenge bit  $c_1$  and  $P_2$  replies with  $c_1 \oplus s_1$ .  $P_1$  measures the time between sending  $c_1$  and receiving the response.  $P_1$  then replies with  $c_2 \oplus s_1$ .  $P_2$  measures the time between sending  $c_2 \oplus s_1$  and receiving the response. This process is repeated  $n$  times. The mutual DB procedure is considered successful if both parties verify all responses and match previously committed values (see [25] for more details). We take advantage of this optimization in constructing mutual GDB protocols.

If prover authentication is required, public key signatures can be used to sign challenges and responses. The verifier validates the signature in the last step, as shown in Figure 1. The protocol succeeds *only* if the signature is valid. Public key identification schemes (e.g., Schnorr or Fiat-Shamir) can also be used as described in [4].



**Figure 2: Group Distance Bounding Variants.**

## 2.2 Problem Statement and System Model

We first present the general GDB problem statement and its variants, then describe our system and adversary models.

**Problem Statement:** In general, GDB involves *one or more provers* interacting with *one or more verifiers*. The goal of verifiers is to accurately and securely establish distance bounds (DBs) to provers and, optionally, authenticate them. Provers are generally untrusted, i.e., they may behave maliciously by reporting false distances and identities. Each device can be a prover, a verifier or both, i.e., we take into account both one-way and mutual DB. We consider three GDB cases (see also Figure 2):

- 1- **MPNV:**  $N$  verifiers establish DBs to  $M$  provers.
  - 2- **IPNV:**  $N$  verifiers establish DBs to a single prover.
  - 3- **MP1V:** A single verifier establishes DBs to  $M$  provers.
- In mutual GDB, the two special cases (1PNV and MP1V) are equivalent and are called (1-to-M). In addition, there is a case where  $N$  peer nodes are required to establish mutual DB with each other; we call it *mutual multi-party GDB*.
- System Model:** We make the following assumptions:
- **Coverage:** All devices are within each others' transmission range. This is a common assumption in all DB literature, e.g., [4, 14, 23, 21, 18, 24].
  - **Accuracy:** Each device can implement distance bounding, i.e., is capable of fast and accurate processing - on the order of nanoseconds<sup>2</sup>.
  - **Keys:** Each device has a public/private key pair and a certificate binding the public key to its identity. (Applies only if authentication is required).
  - **Collusion:** Colluding provers do not reveal their secret keys to each other. (Applies only if authentication is required).
  - **Interaction between Verifiers:** In one-way GDB, verifiers know each others' locations or distances separating them. This is not required in mutual GDB.

**Adversary Model:** We assume that the adversary is computationally bounded and can not prevent nodes within its radio range from receiving its transmissions (i.e., not using directional antennas). In one-way GDB, the adversary can only compromise provers. Verifiers trust each other in one-way GDB. In mutual GDB, all nodes are treated equally with no trust assumptions. Our adversary model covers the following attacks in one-way GDB settings (based on attacks in one-way DB [4]):

- 1- **Distance Fraud Attack:** A dishonest prover claims

<sup>2</sup> Possible using off-the-shelf electronics as in [19] or using UWB ranging platforms e.g.[1, 14].

$DB(s)$	Distance Bound(s)
$P$	Prover
$V$ ( $V_a, V_p$ )	Verifier (subscript denotes active or passive)
$DB_{x,y}$	DB established by verifier $x$ on prover $y$
$t_{x,y}$	Time of flight between nodes $x$ and $y$
$d_{x,y}$	Distance between nodes $x$ and $y$ ( $d_{x,y} = d_{y,x}$ )
$n$ ( $n_a, n_p$ )	Number of DB rounds (subscript denotes active or passive)
$d_a$	Fraction of verifiers performing $n_a$ active rounds
$H()$	Cryptographically secure hash function
$Pr_{ch}(X)$	Fraction of DB rounds in which node $X$ cheats

**Table 1: Notation.**

to be closer than it really is. (Note that a prover can always claim to be further by delaying responses.) The goal of this attack in one-way GDB is to shorten the distance from the malicious prover to one or more (or even all) verifiers.

**2- Mafia Fraud Attack:** A form of a man-in-the-middle (MiTM) attack. The adversary, who is close to the verifier, interacts with it, while posing as the prover. In parallel, it interacts with the prover posing as the verifier. The goal is to fool the verifier into believing that the adversary is the prover located closer to the verifier than the actual prover. In one-way GDB, we consider a version of this attack where the adversary places one or more nodes between the prover(s) and one or more verifiers. The adversary aims to convince verifiers that these intermediate nodes are real provers which are located closer to them than actual provers.

We consider the following attacks in mutual GDB settings:

**1- Passive Distance Fraud Attack:** In mutual GDB with one group of  $N$  nodes, each node has to establish  $N - 1$  DBs. We assume that the adversary can compromise at most  $N - 2$  nodes. The goal of this attack is for two (or more) *uncompromised* nodes to establish incorrect DBs to each other.

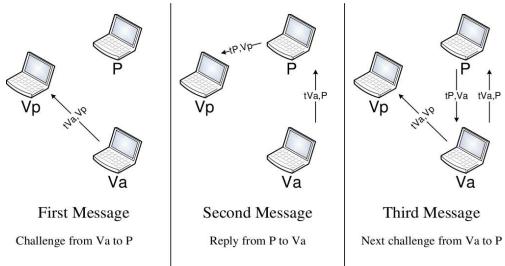
**2- Node Insertion Attack:** The adversary inserts one or more fake nodes into the group. It succeeds if other "honest" nodes in the group accept such fake nodes as legitimate group members and establish DBs to them. Such DB should also be shorter than the real distance to these fake nodes.

## 3. GDB BUILDING BLOCKS

We first introduce a new building block primitive for constructing secure and efficient GDB protocols, *one-way passive DB*. We then consider an optimization to decrease number of messages in GDB protocols, *interleaved one-to-many mutual DB*. By combining the two we construct the novel *mutual multi-party GDB* protocol. In mutual multi-party GDB each node, in a group of  $N$  nodes, engages in a secure mutual DB protocol with its  $(N - 1)$  peers. Notation used in this paper is reflected in Table 1.

### 3.1 One-Way Passive DB

Whenever a prover and a verifier engage in a DB protocol, some information about their locations and mutual distance is leaked [18]. We use this observation in the presence of multiple verifiers. We show that it is unnecessary for *every verifier* to directly interact with the prover ( $P$ ) to establish



**Figure 3: Messages Observed by Passive Verifier.**

a DB. If *at least one* active verifier ( $V_a$ ) interacts with  $P$ , any other passive verifier ( $V_p$ ) can deduce the DB between itself and  $P$  by observing messages between  $P$  and  $V_a$ . We assume that  $V_p$  and  $V_a$  trust each other, know the distance separating them (or each other's locations) and are both required to establish a DB to  $P$ . We address passive DB with untrusted verifiers in Section 5.3.

Figure 3 shows how  $V_p$  observes timings ( $T_i$ ) of messages exchanged in a DB protocol between  $P$  and  $V_a$ .  $V_p$  can construct the following equations:

$$T_1 = t_0 + t_{V_a, V_p} \quad (1)$$

$$T_2 = t_0 + t_{V_a, P} + \alpha_P + t_{P, V_p} \quad (2)$$

$$T_3 = t_0 + 2 \cdot t_{V_a, P} + \alpha_P + \alpha_{V_a} + t_{V_a, V_p} \quad (3)$$

where  $\alpha_P$  and  $\alpha_{V_a}$  are processing times of  $P$  and  $V_a$ , respectively (ideally  $\alpha_P$  is equal to zero) and  $t_0$  is the protocol starting time.  $V_p$  can determine time of flight for signals between  $P$  and  $V_a$  thus computing the distance between them:

$$d_{V_a, P} = c \cdot t_{V_a, P} = c \cdot \frac{(T_3 - T_1) - \alpha_P - \alpha_{V_a}}{2} \quad (4)$$

Where  $c$  denotes speed of light. For  $V_a$  (and  $V_p$ ) to measure the distance between itself and  $P$ ,  $\alpha_P$  must be negligible (or constant) and known<sup>3</sup>.

*Overview of establishing a passive DB:*  $V_p$  uses time difference of arrival (TDoA) of three messages, its own location and  $V_a$ 's location to construct the locus of  $P$ 's possible locations (a hyperbola similar to other TDoA techniques [12]).  $V_p$  then determines the distance between  $V_a$  and  $P$  (as shown in Equation 4) and constructs a circle with a radius equal to that distance. This circle intersects with  $P$ 's location locus at two points ( $s_1$  and  $s_2$ ).  $V_p$  computes DB to  $P$  as the distance between itself and  $s_1$  (or  $s_2$ )<sup>4</sup>.

*Details of establishing a passive DB:* We now demonstrate the details of the procedure. We also show that if  $P$  manages to cheat and shorten the passive DB established by  $V_p$ , then the active DB established by  $V_a$  must also be shortened. Since the DB established by  $V_a$  can not be shortened,

<sup>3</sup>Common assumption in DB literature, e.g., [4, 14, 23, 21, 18, 24].

<sup>4</sup>If  $V_p$  does not know  $V_a$ 's exact location but only the distance to  $V_a$ , then, instead of a sector of a circle,  $V_p$  obtains an area between two circles with radii corresponding to furthest and closest points to  $V_p$  on the hyperbola. In that case the larger radius will be used as a DB to  $P$ .

a passive DB is as secure as the active DB established between  $V_a$  and  $P$ . Suppose  $V_a$  is located at  $(x_a, y_a)$  and  $V_p$  is at  $(x_p, y_p)$ .  $V_p$  knows its own location and that of  $V_a$  (hence the distance  $d_{V_a, V_p}$ ). Without loss of generality we assume  $(x_a, y_a) = (0, 0)$  to be the origin of a coordinate system. It follows that:

$$d_{V_a, V_p} = \sqrt{(x_p - x_a)^2 + (y_p - y_a)^2} = \sqrt{(x_p)^2 + (y_p)^2} \quad (5)$$

We further assume that  $P$  is at  $(x, y)$ .  $V_p$  also knows that:

$$d_{V_p, P} = \sqrt{(x - x_p)^2 + (y - y_p)^2} = \sqrt{(x)^2 + (d_{V_a, V_p} - y)^2} \quad (6)$$

$$d_{V_a, P} = \sqrt{(x - x_a)^2 + (y - y_a)^2} = \sqrt{(x)^2 + (y)^2} \quad (7)$$

If three messages as in Figure 3 are received at times:  $T_1, T_2$  and  $T_3$ , respectively,  $V_p$  computes  $d_{V_a, P}$  as shown in Equations 4.  $V_p$  also computes:

$$c \cdot (T_2 - T_1) = c \cdot \delta_1 = d_{V_a, P} + c \cdot \alpha_P + d_{P, V_p} - d_{V_a, V_p} \quad (8)$$

Where  $c$  is speed of light. However, since  $d_{V_a, P}$  (Equations 4) and  $d_{V_a, V_p}$  (verifiers know distances between them) are known,  $V_p$  obtains:

$$\Gamma = c \cdot (\delta_1 - \alpha_P) + d_{V_a, V_p} = d_{V_a, P} + d_{V_p, P} = \sqrt{(x)^2 + (y)^2} + \sqrt{(x)^2 + (d_{V_a, V_p} - y)^2} \quad (9)$$

Which yields the following formula for the locus of  $P$ 's possible location (which lies on a hyperbola due to TDoA [12]):

$$y = \frac{d_{V_a, V_p} \sqrt{(d_{V_a, V_p}^2 - \Gamma^2)} \pm \Gamma \cdot \sqrt{(4x^2 + d_{V_a, V_p}^2 - \Gamma^2)}}{2\sqrt{(d_{V_a, V_p}^2 - \Gamma^2)}} \quad (10)$$

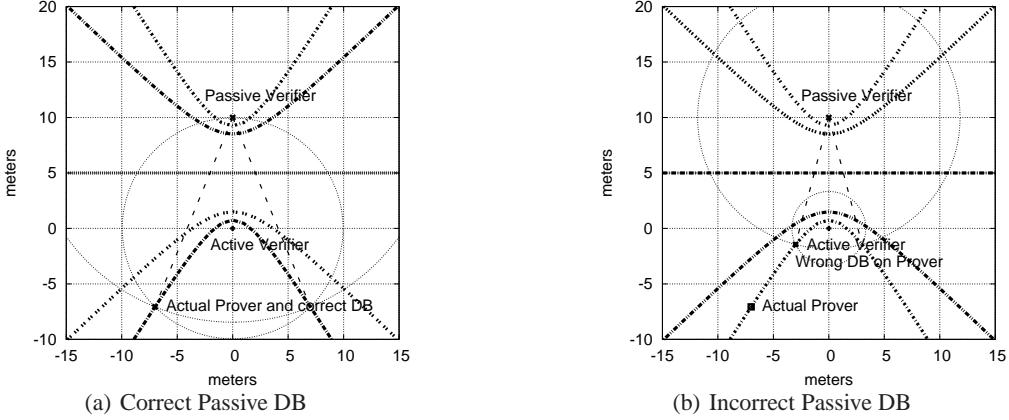
Note that  $DB_{V_a, P} = d_{V_a, P}$  is an upper bound on the distance between  $P$  and  $V_a$ . Using  $d_{V_a, P}$ ,  $V_p$  can construct another equation for the locus of  $P$ 's possible location (a circle around  $V_a$  with radius  $d_{V_a, P}$ ):

$$(x - x_a)^2 + (y - y_a)^2 = (d_{V_a, P})^2 \quad (11)$$

$V_p$  can now establish a passive DB using the intersection of both loci (i.e., solving both equations 11 and 10). This DB is the distance between  $V_p$ 's own location  $(x_p, y_p)$  and the intersection of  $P$ 's loci described by equations 11 and 10. This DB ( $DB_{V_p, P} = d_{V_p, P}$ ) will only be in a sector of a circle, not in the entire circle as in the case of an active DB.

Substituting  $x = x_a + \sqrt{(d_{V_a, P})^2 - (y - y_a)^2}$  (from equation 11) into equation 10, the y-coordinate of  $P$ 's location becomes:  $y \propto (d_{V_a, P})$  (same for  $P$ 's x-coordinate). For  $V_p$  to compute a wrong (shorter) DB to  $P$ , it has to have computed a shorter  $d_{V_a, P}$ . A shorter  $d_{V_a, P}$  requires  $DB_{V_a, P}$  to have been computed shorter than the actual distance between  $P$  and  $V_a$  (which is not possible as shown in Section 2.1).

To better illustrate this, Figures 4(a) and 4(b) show an example scenario.  $P$  (labeled Actual Prover in Figures) at



**Figure 4: Establishing a Correct and Incorrect Passive DB.**

$(-7, -7)$  is on one of several possible hyperbolas. The DB from  $V_a$  at  $(0, 0)$  to  $P$  is shown as a circle around  $V_a$  (labeled as Active Verifier in Figures). If  $P$  somehow cheats so that the passive DB is shorter, then this would require that the circle drawn around  $V_p$  at  $(0, 10)$  intersects the hyperbola at a point  $((-3, -1.5)$  in Figure 4(b)) close to  $V_p$ . This point will be inside the circle established by  $V_a$ . If this is the case then the DB computed by  $V_a$  has to be shorter than the actual distance to  $P$  which is not possible. If  $V_p$  actively engages in a DB protocol with  $V_a$ , it would get the circle shown around it in Figure 4(a). However, in this passive case, it gets a sector of that circle, which is the arc connecting the two points  $((-7, -7)$  and  $(7, -7)$ ) where the computed hyperbola intersects the circle around the active verifier. We have shown in Section 2.1 how active DB prevents the distance fraud attack. Since passive DB is as secure as active DB, it will prevent the distance fraud attack. Adding authentication to passive DB prevents the mafia-fraud attack because an attacker will not be able to authenticate itself to a passive verifier unless it also does to an active one. A passive verifier can utilize the same authentication mechanism as an active verifier. Active verifiers can use public key signatures (or public key identification schemes) to authenticate provers, as described in Section 2.1. All necessary information (commitment, challenges, responses and signatures) required to authenticate provers also reach passive verifiers. The only disadvantage is that a passive verifier does not send its own challenges. Passive DB remains secure because it assumes trusted active verifiers. If that is not the case, mutual one-to-many DB or mutual multi-party GDB can be used.

### 3.2 Interleaved One-to-Many Mutual DB

When one node engages in mutual DB with  $M$  other nodes, *one-to-many mutual DB*, the number of required messages can be reduced by interleaving challenges and responses to different nodes. We label the “one” node in this case the initiator ( $P_i$ ) and the other “many” nodes ( $M$ ) the “participants” ( $P_j$ ,  $j \in \{1, \dots, M\}$ ).  $P_i$  performs mutual DB with each  $P_j$ ;

however, the last message of the interaction with one  $P_j$  is used as the first challenge of the interaction with  $P_{j+1}$ . This process can be generalized for  $M$  parties, one initiator and  $n$  rounds, resulting in a protocol with  $n \cdot (2M + 1)$  messages. This would have required  $n \cdot (4M)$  or  $n \cdot (3M)$  messages if pairwise single prover single verifier DB or interleaved single prover single verifier DB were used respectively.

### 3.3 Mutual Multi-Party GDB

The obvious approach to establish mutual DBs between every pair of nodes, in a group of  $N$  nodes, is to perform it sequentially between each pair. This requires  $2n \cdot N \cdot (N - 1)$  messages and is insecure. A malicious node, acting as a prover, can selectively delay messages to another specific node acting as a verifier. This yields a larger DB, to that node only, and results in false localizations if multi-lateration is used as shown in [8]. One can interleave challenges and responses to reduce the number of messages to  $\frac{(2n+1) \cdot N \cdot (N-1)}{2}$ , but selective delaying of responses will still be possible. Our protocol, *mutual multi-party GDB*, relies on the broadcast nature of the wireless channel and takes advantage of message overhearing and appropriate timing of challenges and responses. *All* nodes simultaneously engage in the *same protocol*. The protocol combines passive DB and interleaving of challenges and responses (similar to Section 3.2) to reduce message complexity from  $O(N^2)$  to  $O(N)$  without sacrificing security.

We begin with a simple four-node example, shown in Figure 5. Each node ( $k$ ) first generates  $n$  random bit strings  $(b_{i,k})$ , each of length  $l$ . Each node broadcasts a commitment to these bit strings. These commitments are hashed and used by nodes to order themselves in a logical ring. This ordering determines the sequence in which nodes send and respond to challenges. In Figure 5 nodes order themselves clockwise starting from  $P_1$  to  $P_4$ .  $P_1$  starts and sends the first of its generated bit strings  $(b_{1,1})$  as a challenge to its left logical neighbor  $P_2$  (message 1).  $P_2$  computes and sends the reply  $(rp_{2,1})$  to  $P_1$  using its own first bit string  $(b_{1,2})$ .

and the challenge bit string ( $b_{1,1}$ ) received from  $P_1$ , i.e., message 2 is  $rp_{2,1} = b_{1,1} \oplus b_{1,2}$ .  $P_3$  uses the reply from  $P_2$  to  $P_1$  as a challenge and computes its own reply to  $P_2$  ( $rp_{3,2} = b_{1,3} \oplus rp_{2,1}$ ) and broadcasts it as message 3.  $P_4$  uses this reply ( $rp_{3,2}$ ) as a challenge from  $P_3$  and computes its own reply ( $rp_{4,3} = b_{1,4} \oplus rp_{3,2}$ ) and sends it to  $P_1$  as message 4.  $P_1$  then replies to  $P_4$  with message 5 containing  $rp_{1,4} = b_{2,1} \oplus rp_{4,3}$ . The process is repeated again counter-clock wise but using the second bit string. All challenges and responses are broadcast and all nodes receive and record them. Nodes only respond to challenges from their immediate logical neighbors. Once a node computes all required DBs, it broadcasts them with a hash of all received challenges and responses (and optionally signs them if authentication is required). This will require four additional messages. We note here that each node independently computes DBs to other nodes based on linear equations constructed from the reception times as illustrated in Linear equations can be solved with standard automated methods, e.g., Gauss elimination or Gauss-Jordan elimination. Table 2. Nodes *do not* rely on any reported measurements from other nodes, hence the same model of distrusting a node acting as a prover as in the original DB protocol holds.

Any mutual DB protocol for four nodes will require four commit (and four de-commit) messages which are not shown in Figure 5. The main difference is in number of messages in the rapid bit exchange phase. In Figure 5 a total of 8 messages are required in that phase, in the case of sequential pairwise DB 24 will be needed and 18 in case of sequential DB with interleaving. The process can be generalized to the case of  $N$  nodes. The total number of messages for the general case of  $N$  nodes and  $n$  rounds in the rapid bit exchange phase is<sup>5</sup>:  $n \cdot (2N)$ . Additionally,  $2N$  messages are required for the commitments and decommitments to make sure that every node has used the random bits it generated and has computed DBs correctly.

*Security:* The mutual multi-party GDB protocol is secure against distance-fraud and passive distance fraud attacks as long as the group contains at least two honest neighboring nodes (in the logical ring). A malicious node launching any attacks will be detected by these two (or more) honest neighbors because the active DB between them can not be influenced by *any* other node. When immediate neighbors of a node exchange messages with their own neighbors, that node establishes passive DBs on these two-hop neighbors. These DBs are established passively and can not be affected by any other node because, as we have shown, passive DB is as se-

<sup>5</sup>This can be derived by analyzing the construction of linear equations from observing messages. Any node can construct  $2N - 2$  independent equations from time of arrival of  $2N$  consecutive messages (as each node sends two messages). These equations have  $2N - 2$  unknowns,  $N$  unknowns for time of flight between pairs of neighboring nodes and  $N - 3$  between the observing node and every other node (see in Figure 6). There's an additional unknown, the variable  $t_0$ , corresponding protocol starting time. These equations can be solved resulting in a unique solution.

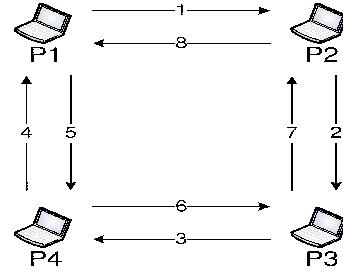


Figure 5: Mutual Multi-Party GDB Example

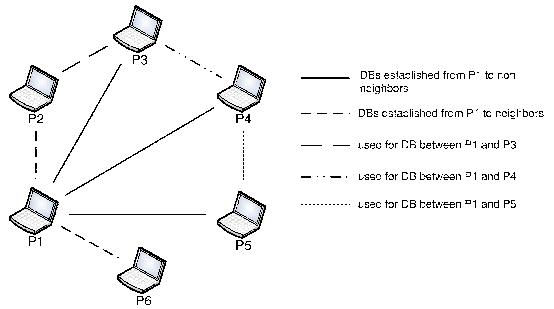


Figure 6: Breaking down Mutual Multi-party GDB into Passive DBs.

cure as active DB. This process is repeated until all DBs are established. The example in Figure 6 shows how node  $P_1$  uses interactions between different nodes in the group to establish a DB on each of them.  $P_1$  establishes DB directly with its neighbors  $P_2$  and  $P_6$ , it then uses the messages exchanged between  $P_2$  and  $P_3$  to establish a passive DB on  $P_3$  and those between  $P_3$  and  $P_4$  to DB  $P_4$  and between  $P_4$  and  $P_5$  to DB  $P_5$ . This process is carried out by each node independently during the protocol at different times resulting in secure DBs established to other nodes. The description of Figure 6 is simplified to convey the intuition. In reality, when solving linear equations constructed from TDoA of messages, several equations resulting from different interactions will be used in computing each DB to a non neighbor (details are shown in Table 2). For example, in Table 2,  $P_1$  uses equations of  $T_7$ ,  $T_3$  and  $T_9$  to DB  $P_3$ , as opposed to  $T_3$  only as in the simplified explanation (Figure 6).

To demonstrate how attacks can be detected consider how each node computes DBs from the arrival times of messages as shown in Table 2. Assuming  $P_1$ ,  $P_2$  and  $P_4$  are honest and  $P_3$  is malicious.  $P_3$  can launch attacks by delaying its messages (number 3 and 7) by  $\delta_1$  and  $\delta_2$  respectively. This attack will be detected because  $DB_{P_1, P_2}$  computed by  $P_1$ , and that computed by  $P_2$  will not be the same. This will be detected when nodes broadcast their computed DBs at the end of the protocol.  $P_1$  will compute  $DB_{P_1, P_2} = T_2/2 = t_{P_1, P_2}$  (from message 2 in the column for  $P_1$ ), whereas  $P_2$  will compute  $DB_{P_1, P_2} = T_5 - t_{P_2, P_3} - t_{P_3, P_4} - t_{P_1, P_4} = t_{P_1, P_2} - (\delta_1 + \frac{\delta_2}{2})$  (from messages 5 and 3 and steps (a) and (b) in the last row of the column for  $P_2$ ). A similar detection

Msg	Participant 1 ( $P_1$ )	Participant 2 ( $P_2$ )	Participant 3 ( $P_3$ )	Participant 4 ( $P_4$ )
1	Sender	$T_1 = t_0 + t_{P_1, P_2}$	$T_1 = t_0 + t_{P_1, P_3}$	$T_1 = t_0 + t_{P_1, P_4}$
2	$T_2 = 2t_{P_1, P_2} \rightarrow DB_{P_1, P_2}$	Sender	$T_2 = t_0 + t_{P_1, P_2} + t_{P_2, P_4} + t_{P_2, P_3}$	$T_2 = t_0 + t_{P_1, P_2}$
3	$T_3 = t_{P_1, P_2} + t_{P_2, P_3} + t_{P_3, P_1}$	$T_3 = 2t_{P_2, P_3} \rightarrow DB_{P_2, P_3}$	Sender	$T_3 = t_0 + t_{P_1, P_2} + t_{P_2, P_3} + t_{P_3, P_4}$
4	$T_4 = t_{P_1, P_2} + t_{P_2, P_3} + t_{P_3, P_4} + t_{P_4, P_1}$	$T_4 = t_{P_2, P_3} + t_{P_3, P_4} + t_{P_2, P_4}$	$T_4 = 2t_{P_3, P_4} \rightarrow DB_{P_3, P_4}$	Sender
5	Sender	$T_5 = t_{P_2, P_3} + t_{P_3, P_4} + t_{P_4, P_1} + t_{P_1, P_2}$	$T_5 = t_{P_3, P_4} + t_{P_4, P_1} + t_{P_1, P_3}$	$T_5 = 2t_{P_4, P_1} \rightarrow DB_{P_4, P_1}$
6	$T_6 = 2t_{P_1, P_4} \rightarrow DB_{P_1, P_4}$	$T_6 = t_{P_2, P_3} + t_{P_3, P_4} + 2t_{P_4, P_1} + t_{P_4, P_2}$	$T_6 = 2t_{P_3, P_4} + 2t_{P_4, P_1}$	Sender
7	$T_7 = t_{P_1, P_4} + t_{P_3, P_4} + t_{P_3, P_1}$	$T_7 = 2t_{P_2, P_3} + 2t_{P_3, P_4} + 2t_{P_4, P_1}$	Sender	$T_7 = 2t_{P_4, P_3} \rightarrow DB_{P_4, P_3}$
8	$T_8 = t_{P_1, P_4} + t_{P_3, P_4} + t_{P_2, P_3} + t_{P_1, P_2}$	Sender	$T_8 = 2t_{P_3, P_2} \rightarrow DB_{P_3, P_2}$	$T_8 = t_{P_4, P_3} + t_{P_3, P_2} + t_{P_2, P_4}$
	(a) $T_7 + T_3 - T_4 = 2t_{P_1, P_3} \rightarrow DB_{P_1, P_3}$	(a) $T_6 - T_4 = 2t_{P_1, P_4}$ (b) $T_7 - T_3 + T_4 - T_6 = 2t_{P_3, P_4}$ (c) $T_6 - t_{P_2, P_3} - t_{P_3, P_4} - 2t_{P_1, P_4} = t_{P_2, P_4} \rightarrow DB_{P_2, P_4}$ (d) $T_5 - t_{P_2, P_3} - t_{P_3, P_4} - t_{P_1, P_4} = t_{P_1, P_2} \rightarrow DB_{P_2, P_1}$	(a) $T_6 - 2t_{P_3, P_4} = 2t_{P_1, P_4}$ (b) $T_5 - t_{P_3, P_4} - t_{P_1, P_4} = 2t_{P_1, P_3} \rightarrow DB_{P_1, P_3}$	(a) $T_8 + T_2 - T_3 = 2t_{P_4, P_2} \rightarrow DB_{P_4, P_2}$

**Table 2: Message Reception Times and Constructed Equations in the Mutual Multi-Party GDB Protocol for Figure 5** ( $t_{i,j} = t_{j,i}$ ,  $t_{x,y} \rightarrow DB_{x,y}$  means that  $DB_{x,y}$  can be directly computed from  $t_{x,y}$ , the last row shows additional computation required to establish the DBs).

will occur between  $P_2$  and  $P_4$  but based on  $DB_{P_2, P_4}$ . Even if  $P_3$  and  $P_4$  are both malicious and colluding, the attack will be detected because  $DB_{P_1, P_2}$  computed by  $P_2$  will be  $DB_{P_1, P_2} = t_{P_1, P_2} - \frac{\delta_2}{2}$  (assuming  $P_3$  delays its messages by  $\delta_1$  and  $\delta_2$  respectively, whereas  $P_4$  delays its message by  $\delta_3$  and  $\delta_4$ ). Variations in computed DBs can be detected if at least two honest nodes are neighbors in the constructed ring.

Node authentication in this protocol can be achieved using traditional public key signatures. Each node initially broadcasts its public key certificate in the commitment phase. Once all  $(2nN)$  protocol rounds are completed, each node hashes all exchanged challenges and responses and signs the resulting hash. Recall that all nodes receive all challenges and responses due to wireless broadcast. All signatures are then broadcasted and each node verifies  $N - 1$  signatures. All nodes are authentic if all signatures verify successfully.

## 4. DB EXTENDED TO GROUP SETTINGS

We now show how to construct protocols for the two most general GDB cases: (1)  $M$  provers and  $N$  verifiers (MPNV) in one-way GDB, and (2) NtoM in mutual GDB. All other cases can be obtained by setting the values of  $N$  and  $M$  as desired. For comparison, we consider a basic GDB protocol where nodes sequentially engage in a naïve single prover single verifier (mutual) DB. In each case we propose an alternative approach based on passive DB, mutual multi-party GDB or one-to-many mutual DB. We assume that  $n$  rounds of DB are required in all cases.

### 4.1 One-Way MPNV GDB

In this case nodes either act as provers or as verifiers. The goal at the end of the protocol is for all  $N$  verifiers to have DBs to all  $M$  provers. In a naive MPNV protocol, each prover interacts sequentially in  $n$  rounds of DB with each verifier. This is repeated until *all* provers have

interacted with all verifiers. The total number of messages is:  $(2n \cdot N \cdot M)$ . When constructing a protocol for this group setting based on passive DB there are two parameters to consider: (1) the number of active and passive DB rounds performed by each verifier and (2) how the active verifiers are selected (i.e., deterministic or probabilistic). The second parameter does not affect the challenges and responses and how every node performs DB but has an effect on the security if verifiers are compromised (discussed in Section 5). Active verifiers can be selected randomly or by any leader election protocol (e.g., [16]), the rest will be passive verifiers. Other strategies could be explored but are out of scope of this paper. The number of active verifiers, active rounds and how many perform passive rounds affects the number of messages required, the time needed for completing the process and the security of the DB. If all verifiers are treated equally two parameters can be used to describe a general protocol: (1) the number of active verifiers and (2) the number of active rounds by each verifier. If each of the  $N$  verifiers is required to perform  $n$  rounds of DB, we call the number of active rounds  $n_a$  (and passive rounds  $n_p = n - n_a$ ). We denote with  $d_a$  the fraction of verifiers which perform  $n_a$  active rounds. The remaining verifiers perform passive rounds. Each verifier will have  $(d_a \cdot (N - 1) \cdot n_a)$  opportunities to execute passive DB with each of the  $M$  provers. Two interesting cases are obtained by setting  $d_a = 1/N$  and  $n_a = n$ , only one verifier interacts actively with all provers, and by setting  $d_a = 1$  and  $n_a = n/N$  all verifiers interact *equally* with *all* provers. By varying these two parameters ( $n_a$  and  $d_a$ ) one can obtain a protocol with the required security level and less messages than sequential pairwise interaction (performance and security analysis in Section 5).

### 4.2 Mutual NtoM GDB

In the general case of *NtoM* mutual DB there are two

Setting	Base Case Number of Messages	Our Protocol Number of Messages
MPNV	$2n \cdot N \cdot M$	$(2n_a + 1) \cdot (N \cdot d_a) \cdot M$
IPNV	$(2n + 1) \cdot N$	$(2n_a + 1) \cdot N \cdot d_a$
MP1V	$(2n + 1) \cdot M$	$2n + \sum_{j=1}^{M-1} (j + 1) \cdot (n - ((M - 1) - j))$
ItoM	$4n \cdot M$	$n \cdot (2M + 1)$
NtoM	$4n \cdot N \cdot M$	$2n \cdot (N + M)$

**Table 3: Number of Messages in GDB Protocols.**

groups,  $G_1$  and  $G_2$ , of  $N$  and  $M$  nodes respectively. All nodes in  $G_1$  are required to establish DBs to *each of* the nodes in  $G_2$  and vice versa, i.e., there is a total of  $2N \cdot M$  (but  $N \cdot M$  unique) DBs to be established. There are three approaches to construct GDB protocols for this setting: (1) based on one-way passive DB, (2) mutual multi-party GDB or (3) one-to-many mutual DB.

(1) *NtoM Using Passive DB*: A fraction ( $d_1$ ) of nodes in  $G_1$ ,  $d_1 \cdot N$ , will establish  $n_{a1}$  active and  $n_{p1}$  passive DB rounds with each of the  $M$  nodes in  $G_2$ . The rest of the  $(1 - d_1)N$  nodes in  $G_1$  establish *only* passive rounds. This step involves one-way DB so at the end *only* nodes in  $G_1$  will establish DBs to nodes in  $G_2$ . Nodes in  $G_2$  are required to perform a similar step where  $d_2 \cdot M$  nodes establish  $n_{a2}$  active and  $n_{p2}$  passive rounds of DB with each of the  $N$  nodes in  $G_1$ . The rest of the  $(1 - d_2)M$  nodes in  $G_2$  establish *only* passive rounds. After this step all nodes in  $G_2$  will have established one-way DBs to nodes in  $G_1$ . This protocol requires nodes in  $G_1$  to trust each other and know each other's locations or distances separating them (same for  $G_2$ ).

(2) *NtoM Using Mutual Multi-Party GDB*: All the nodes in both groups can be regarded as one group of size  $N + M$ . The  $N + M$  nodes can engage in a mutual multi-party GDB protocol as shown in Section 3.3, i.e., the general case of the example of Figure 5. Such a protocol will require  $2n(N + M)$  messages. At the end each node will have DBs to all the  $N + M - 1$  other nodes. Some of these DBs are not required, since we assumed that nodes in  $G_1$  (and  $G_2$ ) don't perform DB on other nodes in the same group.

(3) *NtoM Using One-to-Many Mutual DB*: Each of the  $N$  nodes in  $G_1$  engages in a one-to-many mutual DB protocol described in Section 3.2 with all  $M$  nodes in  $G_2$ . This is a one-to-many mutual DB, so all nodes in  $G_2$  will also establish a DB to each node in  $G_1$ . The total number of messages in such a protocol will be:  $nN \cdot (2M + 1)$ .

## 5. PERFORMANCE AND SECURITY ANALYSIS

We first analyze performance of proposed GDB protocols. We then consider security of active DB in group settings, passive DB with untrusted verifiers and their combination. Our GDB protocols either use mutual multi-party GDB or a combination of passive and active DB. Their security can be understood by analyzing the underlying mechanisms and combinations thereof. Correctness and security of passive DB and mutual multi-party GDB are analyzed in Sections

3.1 and 3.3 respectively.

### 5.1 Performance of GDB Protocols

Table 3 compares number of messages required in one-way and mutual GDB protocols to the base case (running pairwise DB between nodes). Table 4 shows total time required to compute all DBs. We compare against this base case because there are no previous proposals for GDB. Our *NtoM* protocol in both tables is based on mutual multi-party GDB. Our proposals require fewer messages and depend on the fraction of active verifiers and active rounds performed. In the MPNV case, only  $(n_a \cdot d_a)$  messages are required, where  $n_a$  is the fraction of active rounds performed by the fraction of active verifiers,  $d_a$ . Figure 7(c) shows how the number of messages increases as a function of the fraction of active rounds and active verifiers ( $M=10$  provers and  $N=10$  verifiers and  $n=10$  DB rounds). Figure 7(d) shows how the number of messages varies with the number of provers and verifiers (to illustrate this dependency we assume that number of provers is the same as number of verifiers,  $N = M$ , and that fraction of active rounds is equal to fraction of active verifiers,  $n_a = d_a$ ). In the case of 60 nodes (30 provers and 30 verifiers) if the fraction of active verifiers and active rounds is reduced to 0.8, 33% of messages can be saved. Decreasing this fraction to 0.6 saves more than 55% of messages. Similar savings are also attainable for lower and larger numbers of provers/verifiers.

### 5.2 Security of Active DB vs Mutual Multi-Party GDB

The probability of a single prover successfully cheating a single verifier decreases exponentially with the number of DB rounds ( $n$ ) in an active DB protocol. For  $n$  rounds a prover has  $2^{-n}$  chance to successfully guess all challenge bits and send responses ahead of time. This tricks the verifier into measuring a shorter round trip time of flight<sup>6</sup>. A Verifier in an active DB protocol does not have to trust any other entity. In group settings where each pair of provers/verifiers engage in an active DB protocol, these security guarantees still hold. However, active DB in group settings is insecure if used for localization. When a prover actively interacts with each verifier separately, it can selectively enlarge its distance by delaying messages. Verifiers will incorrectly localize the prover using such DBs. Secure-localization schemes must always require *at least three* verifiers to interact with the prover *simultaneously*. The mutual multi-party GDB protocol achieves this by design. In mutual multi-party GDB *all* nodes participate simultaneously in the *same* protocol and overhear each other's messages. A node can not selectively delay messages to other nodes, it can either delay them to *all* nodes or *none*.

<sup>6</sup> $2^{-n}$  is the probability for Brands-Chaum protocol [4], whereas in some other protocols like Hancke-Kuhn [13], this probability is  $(3/4)^{-n}$ .

Setting	Base Case Time	Our Protocol Time
MPNV	$2n \cdot \sum_{i=1}^N \sum_{j=1}^M t_{V_i, P_j}$	$(2n_a + 1) \cdot \sum_{j=1}^{d_a \cdot N} \sum_{k=1}^M t_{P_k, V_j}$
1PNV	$2n \cdot \sum_{i=1}^N t_{P, V_i}$	$(2n_a + 1) \cdot \sum_{j=1}^{d_a \cdot N} t_{P, V_j}$
MP1V	$2n \cdot \sum_{i=1}^j t_{V_i, P_i}, j \in \{1, M\}$	$(n \cdot \max(t_{V_i, P_i})) + \sum_{i=1}^{M-1} t_{V_i, P_i}$
ItoM	$4n \cdot \sum_{j=1}^M t_{P_i, P_j}$	$2n \cdot \sum_{j=1}^{M+1} t_{P_j, P_{(j+1) \bmod (M+1)}}$
NtoM	$4n \sum_{i=1}^N \sum_{j=1}^M t_{P_i, P_j}$	$2n \cdot (N + M) \max(t_{P_i, P_j}), \forall i, j \in \{1, M\}$

**Table 4: Time Required in GDB Protocols.**

### 5.3 Security of Passive DB with Untrusted Active Verifiers

Passive DB with untrusted verifiers is mainly useful in MANETs, where nodes continuously encounter new peers. Passive DB is secure if the active verifier behaves honestly and is trusted as shown in Section 3.1. This will be the case in a fixed (or mobile) verification infrastructure with prior security association, or under the control of one administrative entity. A malicious active verifier can undermine security of passive DB as follows:

(1) *Reporting a Fake Location (or Distance)*: A passive verifier requires the exact location or the distance to the active one in order to be able to construct the DB as shown in Section 3.1. The passive verifier will wrongfully compute the hyperbola (in Equation 10), if the active verifier reports an incorrect location or distance, leading to a wrong passive DB.

(2) *Sending Early Challenges*: Even if the active verifier reports its location or distance correctly, it can send new challenges prematurely. This leads the passive verifier to believe that the prover is closer than it actually is (as shown in Figure 3).  $V_p$  wrongfully computes the distance  $d_{V_a, P}$  intersecting incorrectly with the hyperbola (as shown in Figure 4(a)), and leading to a wrong DB.

An essential aspect of passive DB is implicit trust that the active verifier is behaving honestly, i.e., not cheating by performing either of the previous two attacks. We devise a metric, the *DB Correctness (DBC)*, to illustrate the effectiveness of passive DB in the presence of such attacks. We define *DBC* for  $n$  rounds of passive DB as follows:

$$DBC = 1 - 2^{n \cdot (Pr_{ch}(V_a) - 1)} \quad (12)$$

where  $Pr_{ch}(V_a)$  is the fraction of rounds in which  $V_a$  cheats. Note that cheating in passive DB is different than in active DB. In active DB,  $P$  is the one cheating, whereas in passive DB we are concerned with the case where  $V_a$  is the one cheating. When  $V_a$  does not cheat in any DB rounds,  $Pr_{ch}(V_a)$  will be 0 and  $DBC = 1 - 2^{-n}$ . When  $V_a$  cheats in all DB rounds  $Pr_{ch}(V_a)$  will be 1 and  $DBC = 0$ . The average correctness ( $DBC_{avg}$ ) in a DB to a given prover, obtained by a passive verifier, in the case of  $N$  active verifiers ( $V_{a_1}, V_{a_2} \dots V_{a_N}$ ) (each engaging in  $n_1, n_2 \dots n_N$  rounds of DB respectively) is computed as the average of individual

*DBC* for each verifier:

$$DBC_{avg} = \frac{N - \sum_{i=1}^N 2^{n_i \cdot (Pr_{ch}(V_a(i)) - 1)}}{N} \quad (13)$$

Figure 7(a) shows how  $DBC_{avg}$  is affected by varying fraction of rounds in which active verifiers cheat (x-axis) and the fraction of cheating active verifiers. In the case considered (10 verifiers and 10 DB rounds) even if 50% of the active verifiers cheat in 50% of their rounds, the DB will be established correctly over 98% of the time. As long as less than half of the active verifiers cheat in less than 90% of their rounds, the DB will be correct more than 70% of the time.

### 5.4 Combined Passive/Active DB Security

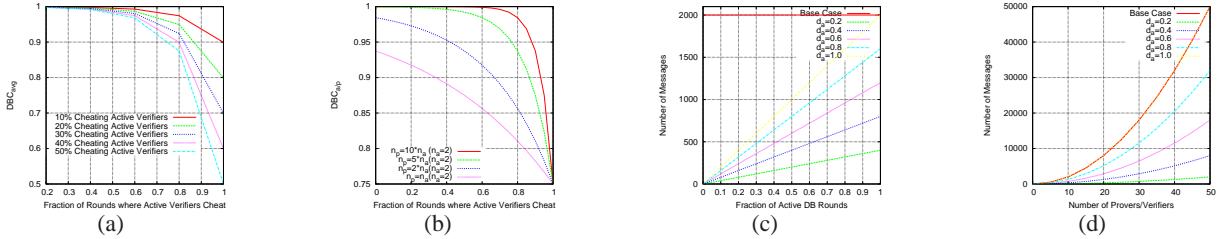
When a verifier performs  $n_a$  active rounds and  $n_p$  passive rounds both can be combined to obtain a more stable DB. We estimate the correctness in such a combined DB using a metric ( $DBC_{a/p}$ ) as follows (note that both passive and active rounds have to result in the same DB):

$$DBC_{a/p} = 1 - (2^{-n_a} \cdot \frac{\sum_{i=1}^N 2^{n_p(i) \cdot (Pr_{ch}(V_a(i)) - 1)}}{N}) \quad (14)$$

If all other active verifiers cheat in all their DB rounds,  $DBC_{a/p}$  becomes that of the active rounds performed by a verifier only, i.e.,  $1 - (2^{-n_a})$ . Otherwise the likelihood of correctness of the established DB increases with any additional passive rounds. Figure 7(b) shows how the DB is affected by cheating of active verifiers during passive DB by showing how  $DBC_{a/p}$  changes. In the case considered (10 verifiers) even if only two rounds of active ( $n_a$ ) DB are performed and as long as the fraction of rounds being cheated in is less than 1 correctness of the DB captured by  $DBC_{a/p}$  increases. Even if the probability of cheating in passive DB rounds is as high as 0.5,  $DBC_{a/p}$  will increase to over 0.95 if there are four or more opportunities to do passive DB.

## 6 RELATED WORK

DB was first proposed in [4] to enable a *single verifier* to determine an upper-bound on the physical distance to a *single prover* and authenticate it as summarized in Section 2.1. Several optimizations and studies of DB were then considered. In particular, [18] studied information leakage in DB protocols as a privacy problem that should be avoided. In our work, we start from this observation to construct passive DB and the mutual multi-party GDB protocol. [25] proposed a mutual DB protocol by interleaving challenges and responses but also between a single prover and a single verifier. [23], [21] and [6] investigated using DB protocols for location verification and secure localization with three verifiers. The setting in [23] is a special case of MPNV with  $M = 1$  and  $N = 3$ . [20] investigated the so-called “in-region verification” and claimed that, for certain applications, such as sensor networks and location-based access control, in-region verification is a better match than location determination. [8] and [7] considered collusion attacks on



**Figure 7: (a)  $DBC_{avg}$  (Equation 13) and (b)  $DBC_{a/p}$  (Equation 14) vs Probability of Cheating in Rounds with Ten Verifiers ( $N=10$ ); (c) and (d) Number of Messages in Passive DB based MPNV Protocol**

DB location verification protocols. Other work, such as [26] looked at using time difference of arrival (TDoA) to determine location of transmitters. [26] proposed using TDoA in the context of Ultra-Wideband (UWB). The work in [24, 14] recently implemented the first RF based TDoA secure localization system using commercial off-the-shelf UWB ranging devices. DB was also studied in the context of ad-hoc networks (e.g., [25]), sensor network (e.g., [15] [6]) and RFID (e.g., [9] [13]) applications. Finally, DB has been used to develop secure proximity based access control protocols for implementable medical devices in [17] and implemented using commercial off-the-shelf electronic components in [19].

To summarize, our work differs from prior results, since: (1) we introduce for the first time *passive DB* and the *mutual multi-party GDB protocol* which are more suitable for group settings, (2) we consider general GDB cases with multiple provers and multiple verifiers (in the one-way and mutual DB settings), (3) we study a large spectrum of possible protocol designs and (4) consider node authentication in both one-way and mutual GDB.

## 7. DISCUSSION AND CONCLUSION

This paper presents the first investigation of group distance bounding (GDB). GDB is a fundamental mechanism for secure operation in wireless networks where verifying distances between, or locations of, groups of nodes is required. We have shown how to construct protocols that are more efficient and secure than applying existing DB techniques in group settings. We made minimal assumptions about GDB settings to make our proposals as general as possible. However we acknowledge two open issues: (1) It remains an open question whether a passive verifier can passively establish a DB without knowing the location of (or distance to) an active one, while perhaps knowing other information about distances to other nodes. (2) We have not addressed denial-of-service attacks in group settings (i.e., noisy environments). We note though that single prover single verifier DB in noisy environments has been addressed in both the one-way and mutual cases in [13, 22].

## 8. REFERENCES

- [1] Multispectral Solutions Inc., Urban Positioning System (UPS). <http://www.multispectral.com>.
- [2] RFC1677-Tactical Radio Frequency Communication Requirements for IPng. <http://www.ietf.org/rfc/rfc1677.html>.
- [3] A. Francillion B. Danev and S. Čapkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Cryptology ePrint Archive: Report 2010/332*, 2010.
- [4] S. Brands and D. Chaum. Distance-bounding protocols. In *EUROCRYPT '93*, pages 344–359. Springer-Verlag New York, Inc., 1994.
- [5] E. Callaway and P. Gorday et. al. Home networking with ieee 802.15.4: a developing standard for low-rate wireless personal area networks. *Communications Magazine, IEEE*, 40(8):70–77, aug 2002.
- [6] S. Capkun and J. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE INFOCOM*, 2005.
- [7] N. Chandran, V. Goyal, R. Moriarty, and R. Ostrovsky. Position based cryptography. In *CRYPTO '09*, pages 391–407, Berlin, Heidelberg, 2009. Springer-Verlag.
- [8] Jerry T. Chiang, Jason J. Haas, and Yih-Chun Hu. Secure and precise location verification using distance bounding and simultaneous multilateration. In *ACM WiSec '09*, pages 181–192.
- [9] S. Drimer and S. Murdoch. Keep your enemies close: distance bounding against smartcard relay attacks. In *SS'07: Proceedings of 16th USENIX Security Symposium*, Berkeley, CA, USA, 2007. USENIX Association.
- [10] F. Roesner et al. A. Czeskis. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy*, 0:447–462, 2010.
- [11] O. Chen et al. C. Chen. Gangs: gather, authenticate 'n group securely. In *MobiCom'08*, pages 92–103, New York, NY, USA. ACM.
- [12] Fredrik Gunnarsson. Positioning using time-difference of arrival measurements. In *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.
- [13] G. Hancke and M. Kuhn. An rfid distance bounding protocol. In *SEURECOMM '05*, pages 67–73, Washington, DC, USA, 2005. IEEE Computer Society.
- [14] H. Luecken M. Kuhn and N. Tippenhauer. UWB impulse radio based distance bounding. In *Proceedings of the Workshop on Positioning, Navigation and Communication (WPNC)*, 2010.
- [15] C. Meadows, P. Syverson, and L. Chang. Towards more efficient distance bounding protocols for use in sensor networks. In *Securecomm and Workshops, 2006*, pages 1–5, 28 2006–Sept. 1 2006.
- [16] J. Welch N. Malpani and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *DIALM'00*, pages 96–103.
- [17] K. Rasmussen, C. Castelluccia, T. Heydt-Benjamin, and S. Čapkun. Proximity-based access control for implantable medical devices. In *ACM CCS'09*, 2009.
- [18] K. Rasmussen and S. Čapkun. Location privacy of distance bounding protocols. In *ACM CCS '08*, pages 149–160, 2008.
- [19] K. Rasmussen and S. Čapkun. Realization of rf distance bounding. In *Proceedings of the USENIX Security Symposium*, 2010.
- [20] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security*, New York, NY, USA, 2003. ACM.
- [21] V. Shmatikov and M. Wang. Secure verification of location claims with simultaneous distance modification. In *ASIAN*, 2007.
- [22] D. Singelée and B. Preneel. Distance bounding in noisy environments. In *ESAS 2007*, volume 4572 of *Lecture Notes in Computer Science*, pages 101–115, Cambridge,UK. Springer-Verlag.
- [23] D. Singelée and B. Preneel. Location verification using secure distance bounding protocols. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, Nov. 2005.
- [24] N. Tippenhauer and S. Čapkun. Id-based secure distance bounding and localization. In *In Proceedings of ESORICS (European Symposium on Research in Computer Security)*, 2009.
- [25] S. Čapkun. L. Buttyán and J. Hubaux. Sector: secure tracking of node encounters in multi-hop wireless networks. In *ACM SASN'03*, pages 21–32, New York, NY, USA. ACM.
- [26] D. Young, C. Keller, D. Bliss, and K. Forsythe. Ultra-wideband (uwb) transmitter location using time difference of arrival (toda) techniques. volume 2, pages 1225–1229 Vol.2, Nov. 2003.