Position Based Cryptography*

Nishanth Chandran Vipul Goyal† Ryan Moriarty Rafail Ostrovsky‡

UCLA

{nishanth,vipul,ryan,rafail}@cs.ucla.edu

Abstract

We consider what constitutes *identities* in cryptography. Typical examples include your name and your social-security number, or your fingerprint/iris-scan, or your address, or your (non-revoked) public-key coming from some trusted public-key infrastructure. In many situations, however, **where you are** defines your identity. For example, we know the role of a bank-teller behind a bullet-proof bank window not because she shows us her credentials but by merely knowing her location. In this paper, we initiate the study of cryptographic protocols where the identity (or other credentials and inputs) of a party are derived from its *geographic location*.

We start by considering the central task in this setting, i.e., securely verifying the position of a device. Despite much work in this area, we show that in the Vanilla (or standard) model, the above task (i.e., of secure positioning) is impossible to achieve. In light of the above impossibility result, we then turn to the Bounded Retrieval Model (a variant of the Bounded Storage Model) and formalize and construct information theoretically secure protocols for two fundamental tasks:

- Secure Positioning; and
- Position Based Key Exchange.

We then show that these tasks are in fact *universal* in this setting – we show how we can use them to realize Secure Multi-Party Computation.

Our main contribution in this paper is threefold: to place the problem of secure positioning on a sound theoretical footing; to prove a strong impossibility result that simultaneously shows the insecurity of previous attempts at the problem; and to present positive results by showing that the bounded-retrieval framework is, in fact, one of the "right" frameworks (there may be others) to study the foundations of position-based cryptography.

*Preliminary version appeared in CRYPTO 2009: 391-407. Research supported in part by NSF grants 0716835, 0716389, 0830803.

†Research supported in part by a Microsoft Research Graduate Fellowship and the above NSF grants.

‡Department of Computer Science and Department of Mathematics. Research supported in part by an IBM Faculty Award, Xerox Innovation Group Award, NSF grants 0430254, 0716835, 0716389, 0830803 and U.C. MICRO grant.

1 Introduction

1.1 Motivation

In cryptography, typically a party will possess a set of credentials determining: its identity, what tasks it can do, which protocols it can participate in and so on. These set of credentials will typically correspond to the party having some of the following attributes: some secret information (e.g., a secret key), authenticated information (e.g., a digitally signed certificate from a trusted entity), biometric feature and so on. In this paper, we ask the following question: can the *geographical position* of a party be one of the credentials? The geographical position of a party is valuable in a number of natural settings. We give a few examples:

- **Position based Secret Communication.** Consider communication between different military establishments. For example, the Pentagon in Washington D.C. might want to send a message (having some classified information) such that it can only be read by an individual present at the US military base in South Korea. In a traditional solution, the South Korean military base will have a secret key to decrypt the message. However, the enemy might try to break into the military base computers to capture this key. It would be desirable to add an additional layer of security that would guarantee that anyone reading the message is *physically* present at the South Korean base.
- **Position based Authentication/Signatures.** In the above example, suppose the South Korean military base wants to send some information to the Pentagon. It would be desirable for the Pentagon to have a guarantee that the message was indeed sent from the geographical position of the military base.

Indeed, the above list is not exhaustive. One could think about *position based access control* (where access to a resource needs to be restricted to certain locations, e.g., a printer or fax machine is accessible only to people inside some set of offices) and *pizza delivery* (where the pizza company first wants to verify that the person placing the order is indeed located at the delivery address he specified). To perform such "position specific" tasks, we introduce the notion of *position based cryptography*.

The first natural question that arises is: "Can you convince others about where you are?". More precisely, we have a prover who claims be at a geographical position P. There is a set of remote verifiers (or in other words, a positioning infrastructure) who wish to make sure that the prover is indeed at position P as claimed (for example, by executing a protocol with that prover). We call the above problem as "Secure Positioning". The question of secure positioning is a fundamental one and deals with designing a system which enables a prover to communicate back and forth with a group of verifiers to give them an *interactive proof of its geographic position*.

The problem of secure positioning is well studied in the security community (see e.g., [SSW03, SP05, Bus04, CH05, CCS06]). The de-facto method to perform secure positioning is based on the time of response technique where the messages travel with the speed of radio waves which is equal to the speed of light (*this is similar in nature to how the commercial GPS systems work, see section 1.4*). At a high level, the verifiers will send messages to the device and will measure the time taken to receive a response. Although there have been several proposed protocols for secure positioning, all of them are completely insecure under the so called "collusion attack". That is, if a set of (possibly cloned) provers collude together and work in a controlled manner during the protocol execution, the provers will be able to convince the verifiers that

the verifiers are talking to a prover at position P (even though none of the adversarial provers may be at P). We in fact show that, unfortunately, such an attack is unavoidable. That is, it is impossible to have secure protocols for positioning in this Vanilla model (even if one is willing to make computational assumptions). Hence, we cannot hope to realize most of the meaningful position based tasks.

In light of the above drawbacks, in this paper we explore the intriguing possibility if secure positioning protocols exist which can resist collusion attacks. In search of an answer to this question, we turn to the bounded retrieval model (BRM), which is a variant of the bounded storage model (BSM) introduced by Maurer [Mau92]. Quite surprisingly, this model turns out to be a right model for proving the security of position-based cryptographic tasks. We first construct a protocol for information theoretic secure positioning in this model. To our knowledge, this is the first protocol which is secure even against collusion attacks. Although secure positioning is an important step, the full power of position based cryptography can only be realized if we achieve key exchange with the device at a particular geographic position. Hence we introduce position based key exchange and present two protocols to achieve it in the BRM. Our first protocol achieves security against a computationally bounded adversary (in the BRM). In this protocol, we achieve key exchange between the verifiers and any device at position P that is enclosed within the tetrahedron formed between 4 verifiers in 3-dimensional space. Our second protocol achieves information theoretic key exchange between the verifiers and devices at positions P that lie in a specific geometric region (characterized by a condition that P must satisfy) within the tetrahedron.

Note that we are interested only in verifying the position claim of devices that are within the tetrahedron enclosed between the 4 verifiers. This is not a limitation, since apriori, we are restricting, by geographical bounds, the locations where an honest device can be located (such as inside a room, to get access to a printer or a hard drive). If a device makes a position claim that lies outside of this region, we reject the claim without any verification. We stress, however, that we do not make *any* assumption about the positions of adversaries in the system. In particular, this freedom for the adversarial devices guarantees that no set of adversaries (some of whom may even be outside of the tetrahedron) can falsely prove that any one of them is at position P inside the tetrahedron as long as none of them are at position P.

1.2 The Two Models Considered

The Vanilla Model. We now informally describe the Vanilla model. We have a device (also referred to as the prover) who is located at a position P (where P is a point in a d-dimensional Euclidean space). There exists a set of verifiers $\{V_1, V_2, ..., V_m\}$ at different points in the d-dimensional space, such that P lies inside the tetrahedron enclosed by the verifiers. The verifiers are allowed to execute a protocol with the prover to achieve some task. More precisely, a verifier can send messages to the prover at different points in time (with a speed up to the speed of radio waves) and also record the messages which are received from it (along with the time when they are received). The verifiers have a secret channel among themselves using which they can coordinate their actions by communicating before, during or after protocol execution. There could be multiple adversaries with possibly cloned devices who share a covert channel and collude together. This setting is referred to as the Vanilla model.

The Bounded Retrieval Model. The bounded storage model (BSM) was introduced by Maurer in [Mau92] and has been the subject of much work [Mau92, CM97, CCM98, AR99, Din01, ADR02, DR02, Lu04, Vad04, MSTS04, DHRS04, DM04a, DM04b, Din05]. Very roughly, this model assumes that there

is a bound on the amount of information that parties (including an adversary) can store. It assumes the existence of random strings, having high min-entropy, available to the parties at some point in the protocol. An adversary is allowed to store an arbitrary function of these strings, as long as the length of the output of the function is not longer than the adversary's storage bound. A closely related model to the BSM is the bounded retrieval model (BRM), introduced and studied in various related contexts by Di Crescenzo et al. [CLW06] and Dziembowski [Dzi06a, Dzi06b]. This model assumes that parties can store information having high min-entropy, but an adversary can only retrieve part of it. Recently, Dziembowski and Pietrzak [DP07] introduced *intrusion resilient secret sharing* where shares of a secret (stored on different machines) are made artificially large so that it is hard for an adversary to retrieve a share completely, even if it breaks into the storage machine. We note that in the current work, we build on the techniques of [DP07]. We extend these techniques and combine them with geometric arguments to prove the security of our protocols.

In the context of position based cryptography, by bounded retrieval model, we mean the Vanilla model setting where the verifiers can broadcast information having high entropy (or control a randomness source which can) such that the adversaries can only retrieve, say, a constant fraction of this information as it passes by at high speed. The assumption that the adversaries cannot retrieve all the information that goes by seems very plausible in our setting since the information travels at a very high speed (particularly when, e.g., the verifiers have several sources broadcasting information at different frequencies).

We note that our assumption, in some sense, is actually weaker than what is typically assumed while working in the bounded retrieval model. Unlike in BRM, we do not need to assume that honest parties themselves can store the strings that are being broadcast.

1.3 Our Contributions

In this paper, we give the following results towards developing a theory of position based cryptography:

- We begin with a lower bound for the Vanilla model in Section 3. We show that there does not exist a protocol in the Vanilla model using which a group of verifiers can securely verify the location claim of a prover. The impossibility is obtained via an explicit attack which does not depend on the computational power of the parties. To begin with, the lower bound holds if all the parties (i.e., the verifiers, the honest prover and the adversaries) are given unbounded computational power. Further, it holds even if the verifiers are given unbounded computational power but the adversaries (and thus the honest prover) are restricted to being probabilistic polynomial time (PPT) machines (i.e., one may make cryptographic hardness assumptions). With the impossibility of this most fundamental task, we cannot hope to perform most other meaningful position based tasks (including position based key exchange) in the Vanilla model. Capkun et al. [CCS06] propose a protocol in which they assume that the geographical position of the provers can either be "hidden" (not known to the adversary) or that the provers can move rapidly in geographical space.

- Given the above severe lower bound, the task of now choosing a model in which protocols for secure positioning exist becomes a tricky one. One of the main technical contributions of this paper is to connect the bounded retrieval model to position based cryptography. Remarkably, bringing these seemingly unrelated ideas together enables us to achieve meaningful and elegant protocols and proofs of security for position based cryptography.

- In the BRM, we give a protocol for secure positioning (in Section 5) which is provably secure against any number of (possibly computationally unbounded) adversaries colluding together, as long as the total amount of information they can retrieve and store is bounded. To our knowledge, this is the first protocol for positioning which does not fail against collusion attacks. We also describe, in Section 6, how our protocol for secure positioning can be compiled with an unauthenticated computationally secure key exchange protocol (like Diffie-Hellman) and a non-malleable zero-knowledge protocol to achieve computationally secure position based key exchange in the BRM. That is, only a prover who is at a designated position P will receive the key to be shared (even under the presence of any number of PPT adversaries that can only retrieve a bounded amount of information).

- We then present a protocol (in Section 7) that does information theoretically secure key exchange between the verifiers and a device at P. The construction of such a protocol turns out to be surprisingly intricate. While our secure positioning (and computationally secure position based key exchange) can handle claims of all positions P that lie within the tetrahedron formed between 4 verifiers in 3-dimensional space, our information theoretic key exchange protocol can handle positions P that lie in a specific region (which we characterize, using a geometric argument, by a condition that P must satisfy) within the tetrahedron. In Appendix H, we show (for a few example cases) that this region is a large fraction of the enclosed tetrahedron and also provide some figures containing what this region looks like (for various placements of the 4 verifiers). In order to show the security of our protocol, we need to rely on delicate timing arguments (based on geometric properties) as well as prove that the protocol of [DP07] is secure even in the case when multiple parallel adversaries can gain access to the machines and may collude after they have finished accessing the machines.

- Using the above two fundamental protocols as building blocks, we demonstrate that the protocols for more complex tasks can be readily constructed. We consider the problem of establishing a secure channel between two devices (such that each device has a guarantee on the geographic position of the device at the other end). After establishing pairwise secure channels, a group of devices can perform "position based" multi-party computation, where associated with each input, there is a guarantee about the position of the device giving that input. We also discuss the setup of a position based public key infrastructure, where a certificate provides a guarantee about the position (as opposed to the identity) of the owner of the public key in question (We discuss these applications in further detail in Appendix 8.). We remark that the above is obviously not intended to be an exhaustive list of applications and one can construct protocols for several other tasks.

- Our techniques simultaneously show the inadequacy of previous works in addressing collusion attacks. Our impossibility results show explicit attacks on several previously proposed secure positioning protocols in the colluding adversaries scenario (we however note that many of these works did not aim at addressing colluding adversaries). Furthermore, we study the model proposed in [CCS06] that makes the assumption that there can exist verifiers that are "hidden" (whose geographical position is not known to provers and adversaries). In this model, they provide protocols to achieve secure positioning. Indeed, our impossibility results do not directly apply to this model. However, we show that the protocols in [CCS06] are also susceptible to multiple colluding adversaries, although the attack required is more subtle than in other cases. Our attacks are general and suggest that any protocol in such a model can

be broken against multiple adversaries (we leave generalization of our attacks to impossibility results to future work).

Our results do not require any pre-sharing of data (cryptographic keys and so on) between the prover and the verifiers. The only required credential of a prover is its *real geographic position*.

Open Problem: Other Models for Position Based Cryptography. By turning to the bounded retrieval model, we are able to provide the first provably secure constructions of cryptographic tasks that use position as an identity. Given our strong impossibility results in the Vanilla model, an important open question is: do there exist other natural models that allow us to obtain positive results of similar nature?

1.4 Related Work

Secure Positioning. We remark that the problem of position-based cryptography as such has not been studied before. However, secure positioning is a well-studied problem in the field of wireless security. There have been several proposed protocols ([BC94, SSW03, VN04, Bus04, CH05, SP05, ZLFW06]). All these protocols are susceptible to the collusion attack outlined earlier. One can get around this problem of multiple cloned adversaries by assuming a setup phase where the verifiers give an *unclonable* tamper-proof hardware (having some secret information) to all possible future provers. However in the current work, we focus on the setting where the *only* credential needed by a prover is its geographical position.

In [CCS06], a model is considered, that makes the assumption that there can exist verifiers that are covert or hidden to provers and adversaries. Based on this, they provide solutions to secure positioning. The protocols in [CCS06] are also susceptible to multiple colluding adversaries, although the attack required is more subtle than in other cases. We outline this attack in Appendix C. Chiang et al. [CHH09] directly study the problem of colluding adversaries in the context of secure positioning. They propose a secure positioning protocol which is the "best possible" in the vanilla model. Their results, together with ours, imply that the protocol they propose is secure against a collusion of *two* adversaries. However, there is an explicit attack if the number of colluding adversaries is three or higher.

A detailed description of related work on secure positioning and work in the BSM/BRM is presented in Appendix A.

Global Positioning System. The problem addressed by the global positioning system (GPS) is complementary to the one considered in our work. In GPS, there is device trying to determine *its own* geographic position with the aid of various satellites (in a non-adversarial setting). The GPS satellites continually broadcast information in a synchronized manner with the speed of light. The time taken by the information broadcast by various satellites to reach a GPS receiver enables the receiver to compute its position using triangulation techniques.

2 The Model

In this section, we briefly discuss our model. More details can be found in Appendix B. There are three types of parties in our model: Prover, Verifier and Adversary. We treat time and space as "continuous" (rather than discrete). We assume that messages travel at a speed equal to that of radio waves (which is

the same as the speed of light). In the beginning, each party (prover, verifiers and adversaries) is given as input, party's own position (as a point in the d-dimensional space), the position of all verifiers and the security parameter κ. The verifiers and the adversaries are given the claimed position of the prover.

The parties can send out the following two types of messages : (a) Broadcast messages: A broadcast message originating at a position P travels in concentric hyperspheres centered at P in all directions, (b) Directional messages: A directional message, instead of traveling in all directions, travels only in a specific direction specified by a sector. Such messages can be sent using directional antennas. Additionally, verifiers have a private channel among themselves which allows them to talk to each other secretly. Adversaries also have a private (and covert) channel among themselves which allows them to talk to each other secretly such that no verifier suspects any adversarial activity. More details about these messages (along with formal definitions of secure positioning and key exchange) can be found in Appendix B.

The above is our so called Vanilla Model where we prove the impossibility of realizing the most basic position based task (i.e., secure positioning). We assume that parties can send directional messages in the Vanilla model in order to prove a strong lower bound. As noted earlier, all our positive results are in the BRM. Our bounded retrieval model is the same as the Vanilla model except for the following changes:

- Verifiers "possess" a reverse block entropy source (defined formally in Appendix D) capable of generating strings with high min-entropy, say $(\delta + \beta)n$, where n is the length of the string (and $0 < \delta + \beta < 1$; it is also called min-entropy rate). By possessing a reverse block entropy source, we mean that either the verifier itself is capable of generating such a string of high min-entropy, or it has a randomness source (located at the same point in space as itself) which generates and broadcasts such a string. We do not assume that the verifiers can retrieve and store the broadcasted string of data themselves. Generating a lot of entropy is easy; one can think of an "explosion" which generates a lot of noise that can be measured but not stored.

- There exists a bound βn on the total amount of information the adversaries can retrieve as the information passes at a high speed. The retrieval bound βn could be any constant fraction of the min-entropy $(\delta + \beta)n$. The honest parties (including the verifiers) are required to have a storage capacity of only $O(\kappa \cdot \log(n))$.

- Verifiers and provers cannot send directional messages. We however do not restrict the adversary from sending directional messages.

- Let X be a string having large min-entropy as before. The sender (which is a verifier) generates X and sends it out. Any receiver gets to retrieve and store $f(X)$ (for any arbitrary f) in a way such that the total amount of information which it has retrieved does not exceed the retrieval bounds. In case a party receives multiple strings simultaneously, it can retrieve information from these strings, in any order, any number of times (i.e., we do not restrict the adversaries to retrieve information from a string only once) as long as the total retrieval bound is not violated on the amount retrieved.

Observe that the last step above also enforces that any information about a string X (having large min-entropy) that is sent from one adversary to the other is also bounded (since an adversary gets to retrieve and resend only $f(X)$). This rules out simple "reflection attacks" to create a huge storage (where a pair of adversaries close to each other just keep reflecting the string X to each other hence essentially storing X thus violating the bounded storage assumption).

Relaxing Assumptions. For clarity of exposition during our positive results, we make the assumption that the devices can read bits from the string and perform computations instantaneously. We refer the reader to Appendix B for details on how to remove this assumption.

3 Lower Bound on Secure Positioning in the Vanilla Model

We now show a lower bound for the Vanilla model. We show that there does not exist a protocol in the Vanilla model using which a group of verifiers can securely verify the location claim of a prover. The impossibility is obtained via an explicit attack which does not depend on the computational power of the parties. To begin with, the lower bound holds if all the parties (i.e., the verifiers, the honest prover and the adversaries) are given unbounded computational power. Further, it holds even if the verifiers are given unbounded computational power but the adversaries (and thus obviously the honest party) are restricted to being PPT machines (i.e., one may make cryptographic hardness assumptions). Finally, we present a few extensions of our lower bound in Appendix C.

Theorem 1 *There does not exist a protocol to achieve secure positioning in the Vanilla model.*

Proof. Let there be n verifiers $\{V_1, V_2, ..., V_n\}$ that take part in a protocol to verify that a prover is at a position P . We show that for any protocol, there exists a set of l adversaries (l to be defined later) who can interact with the verifiers in such a manner that it is impossible to distinguish if the verifiers are interacting with an adversary or the actual prover.

Consider the hypersphere of radius r around position P such that the distance between V_i and P for all i be strictly greater than r. In other words, we require that r is such that no verifier is present within the hypersphere of radius r centered at position P. For all i, let the straight line joining V_i and P intersect the hypersphere at position I_i. Let there exist $l \leq n$ such intersection points. We note that l could be less than n because, two (or more) verifiers $V_i, V_j, i \neq j$ may be such that P, V_i and V_j lie on the same straight line in d-dimensional space. We place adversaries $A_1, A_2, ..., A_l$ at points $I_1, I_2, ..., I_l$. The verifiers may run an interactive protocol with the prover in order to verify that the prover is at position P. We show that these l adversaries together can simulate the execution of the protocol in such a way that the verifiers cannot distinguish between an execution in which they are interacting with the prover at P and an execution in which they are interacting with these set of adversaries.

Any verification protocol is a series of messages (along with corresponding times), each being from one of the n verifiers to the prover or vice-versa. The verifiers can then verify the position of the prover by analyzing the message they sent and the response they got (along with corresponding times). We give a strategy for every A_m such that the adversaries together can prove that they are at position P.

Let the time taken for any message to travel between V_i and P be T_i. Note that the distance between A_m, for all m, and P is fixed (equal to r). Hence, let the time taken for a message to travel between A_m (for all m) and P be α. Let the set of verifiers that lie on the same straight line that connects A_m and P be $\mathcal{V}_m$. Let the distance between two adversaries A_m and $A_{m'}$ be $dist(m, m')$ (note that $dist(m, m) = 0$).

Now during the protocol execution, every A_m does the following. A_m only listens to messages sent by all $V_i \in \mathcal{V}_m$ and ignores messages sent by other verifiers. A_m is at a location such that all the messages sent by V_i (s.t., $V_i \in \mathcal{V}_m$) to the point P would be received by it (irrespective of whether V_i sends a broadcast message or a directional message). Lets say that a message M is received from a verifier V_i. For

every adversary $A_{m'}$ (including itself, i.e., $1 \leq m' \leq l$), A_m internally delays M by the duration of time $delay(m, m') = 2\alpha - dist(m, m')$, and then sends it to $A_{m'}$ over the covert channel. Hence, every single adversary (including A_m itself) would receive the message at time 2α (over the covert channel) after the time when A_m receives it from V_i (over the broadcast or directional channel).

For every adversary A_m, now assume that the protocol requires the honest prover to send a reply message, at time t, in directions such that verifiers in set $\mathcal{V}_m$ would receive it (note that since all of them are in a straight line in the same direction of point P, either all of them would receive it or none would). In that case, A_m computes the response message using its view over the covert channel so far and sends it at time $t + \alpha$ using a directional message (such that only verifiers in $\mathcal{V}_m$ receive it). However, A_m does not send any messages to V_i for $V_i \notin \mathcal{V}_m$ (if the verifiers in other sets are required to receive this message as well, they will be "taken care of" by other adversaries near them).

The following simple argument shows that every adversary A_m runs exactly a copy of the execution of the prover, only at a time α later. Once this is shown, since it takes time T_i for a prover to send a response to V_i when $V_i \in \mathcal{V}_m$, and it takes A_m only time $T_i - \alpha$, the exact same response will reach V_i at exactly the same instance of time (irrespective of whether it originated at P or at the location of A_m).

We show that the following two statements are true. $delay(m, m')$ is a non-negative value for all m, m' and every message which reaches the prover at P will reach all the adversaries after exactly time α. This will prove that all adversaries run exactly the same copy of the prover, but at a later instance of time.

The first statement follows trivially from triangle inequality. For the second statement, assume that verifier V_i sends a message to the prover at time t. Let m be such that $V_i \in \mathcal{V}_m$ and t' be the time taken for a message to travel between V_i and A_m. The honest prover clearly receives the message at time $t + t' + \alpha$. The adversary A_m receives the message at time $t + t'$ and hence all the adversaries receive it at time $t + t' + 2\alpha$ over the covert channel.

This proves that all adversaries run exactly the same copy of the prover, but at an instance α later. Hence, any execution of a protocol run between the n verifiers and the prover can be simulated by l adversaries running the protocol with the n verifiers. □

We remark here that the above impossibility result holds even in a stronger model where there is a fixed bound on the number of adversaries, as long as this bound can depend on the number of verifiers in the system (but not on the secure positioning protocol itself). This motivates our search for alternative models, where we do not restrict the number of adversaries and still achieve positive results.

4 Preliminaries

Vadhan [Vad04], introduced BSM pseudorandom generators (PRG). Informally, for string X sampled from a distribution having high min-entropy and for a uniformly random seed K, the distribution of the output of the BSM PRG (denoted by $\text{PRG}(X, K)$), is statistically close to the uniform distribution of appropriate length even when given K and $A(X)$ where A is any arbitrary function with bounded output length. We introduce a relaxation of BSM PRGs, which we call BSM entropy generators (EG). The difference between a BSM EG and a BSM PRG is that the output distribution of a BSM EG is only guaranteed to have high min-entropy, and not necessarily be close to the uniform distribution. We refer the reader to Appendix D for formal details about the definitions, constructions and instantiations.

5 Secure Positioning in the Bounded Retrieval Model

In this section, we propose protocols for secure positioning in the BRM. We shall build upon the primitives described in Section 4. To make the intuition clear, we first give a secure positioning protocol for 1-dimension.

5.1 Secure Positioning in 1-dimension

For 1-dimension, we employ two verifiers, denoted by V_1 and V_2 (which send messages with the speed of radio waves). We assume that the position P being claimed by the prover is located between V_1 and V_2. Our protocol is secure against an arbitrary number of adversaries colluding together to prove a position P, as long as the total information that these adversaries can store during the protocol is bounded. We let βn denote the aforementioned retrieval bound. Verifier V_1 is assumed to possess a random source $X_1, X_2, \cdots$ which is a reverse block entropy source of minimum entropy rate $\delta + \beta$, where $X_i \in \{0,1\}^n$.

We shall use a (ε, ψ)-secure BSM entropy generator EG: $\{0,1\}^n \times \{0,1\}^\ell \rightarrow \{0,1\}^m$ as discussed in the previous section. We choose the input size ℓ such that $\varepsilon + 2^{-\psi}$ is negligible in the security parameter κ. An example of a fast BSM EG, which is just a random sampler requiring no computations at all, is presented in Section D.3. To use the instantiation of EG given in Section D.3, we set $\ell \geq (2/\delta)\kappa log(n)$.

Before the protocol starts, the verifier V_1 selects a key $K \xleftarrow{R} \{0,1\}^\ell$ and sends it to verifier V_2 over the private channel (using a private multicast message). Let t and t' be the time taken for radio waves to reach P from V_1 and V_2 respectively. Verifier V_1 sends out X from the reverse block entropy source such that X has min-entropy $(\delta + \beta)n$. At the same time, V_1 computes $\mathrm{EG}(X, K)$ and stores it on its output tape. Let T be the time at which X reaches P. Verifier V_2 sends the key K out at a time such that it *meets* X at time T at the position P. More precisely, X and K are sent at times $(T - t)$ and $(T - t')$ by V_1 and V_2 respectively.

At time T, the prover claiming to be at position P evaluates $y = \mathrm{EG}(X, K)$ and sends it back to the verifier V_1. Verifier V_1 verifies that the string y is received at time $(T + t)$ and that it equals $\mathrm{EG}(X, K)$. If these verifications succeed, the position claim of the prover is accepted and it is assumed to be indeed at position P. Otherwise, the position claim is rejected.

The protocol clearly satisfies the completeness property since an honest prover at position P will have both X and K available at time T and hence it can compute y (by asking the hypothetical ITM P_{env} to compute the function $\mathrm{EG}(., K)$.) and report it back to V_1 by time $(T + t)$. We discuss the security below:

Theorem 2 *The 1-dimension secure positioning protocol is secure against an arbitrary number of adversaries colluding together, with the total adversary information retrieved bounded by βn.*

Proof. Suppose there exists an adversarial strategy with which a set of adversaries, none of which is at position P, are able to report back the correct y to the verifier V_1 at time $(T + t)$ with a non-negligible probability in the security parameter. We show that the above contradicts the properties of the EG.

We consider the state of the system at time T. X and K are at position P. Let there be g adversaries between V_1 and P and the information they have retrieved about X be $S_1, S_2, ..., S_g$ respectively. Let S denote the combined information $S_1 \cup S_2 \cup ... \cup S_g$. Clearly since K has not yet crossed P, S is an arbitrary function of X alone. Further, $|S| \leq \beta n$ since βn is the total retrieval bound. Now we have the following:

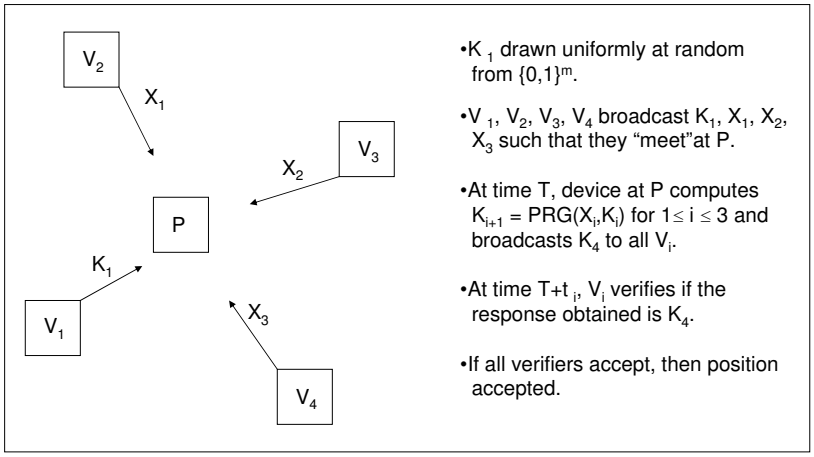

Figure 1: Secure positioning protocol in 3-Dimensions

Lemma 1 *The string y to be sent to the verifier V_i at time $(t+T)$, can be an arbitrary function of S and K alone. More formally, given an adversarial strategy to compute y in our setting, there exists a simulator that outputs y only given S and K (and not the string X).*

The above lemma holds because (a) S is the only information stored by the adversaries between V_1 and P, (b) there is no adversary at P, and, (c) any information about X between P and V_2 at time T cannot reach V_1 by time $(t+T)$.

Hence we have $y = A(S, K)$, where $A(.,.)$ is any arbitrary adversarial algorithm. However, given S and K, using properties of the BSM EG, the probability of an adversary correctly guessing y is upper bounded by $\varepsilon + 2^{-\psi}$. But $\varepsilon + 2^{-\psi}$ is negligible in the security parameter by our choice of ℓ. Thus we have reached a contradiction. □

5.2 Secure Positioning in 3-dimensions

We generalize the above protocol to obtain a protocol for secure positioning in 3-dimensional space. βn is the total adversary information retrieval bound. We use 4 verifiers denoted by $V_1, \cdots, V_4$ possessing reverse block sources of minimum entropy $(\delta + \beta)n$ that output strings X_i. Position P being claimed by the prover is enclosed in the tetrahedron defined by these 4 verifiers. t_i is the time taken for radio waves to reach the point P from verifier V_i. PRG:$\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^m$ is an ε-secure BSM pseudorandom generator. We choose the parameters such that $\varepsilon + 2^{-m}$ is negligible in the security parameter. In order for the verifiers to themselves compute the response expected from the prover, we first assume that verifiers can store the X_i values. We later show how this assumption can be removed. The protocol is illustrated in Figure 1. For further details, refer Appendix E.

The completeness follows from the fact that verifiers can compute K_4 from the stored X_i values and the prover can also compute K_4 since all the information required is present jointly at P at time T. The security of this protocol is proven using techniques from the proof of security of the protocol for 3-dimensional position based key exchange that is discussed in Section 7 (note that position based key exchange implies a protocol for secure positioning).

We now describe, how to remove the assumption that verifiers can store strings drawn from their

respective reverse block entropy sources. Note that the problem we face when verifiers cannot store the large strings is that verifiers have no way of verifying the response of the prover. This is because, when for example, V_3 broadcasts string X_2, it does not know the key K_2 used to compute K_3 from X_2. We get around this problem as follows. The verifiers pre-determine the keys K_1, K_2, K_3, K_4 that are to be used at every iteration of the application of the PRG. Now, the expected response of the prover, K_4 is known before protocol execution to all verifiers. The protocol is as follows:

1. V_1, V_2, V_3 and V_4 pick keys $K_1, K_2, K_3, K_4 \xleftarrow{R} \{0,1\}^m$ and broadcast them over their private channel.
2. V_1 broadcasts key K_1 at time $T - t_1$. V_2 broadcasts X_1 at time $T - t_2$ and simultaneously also broadcasts $K_2' = \text{PRG}(X_1, K_1) \oplus K_2$. Similarly, V_3 broadcasts $(X_2, K_3' = \text{PRG}(X_2, K_2) \oplus K_3)$ at time $T - t_3$ and V_4 broadcasts $(X_3, K_4' = \text{PRG}(X_3, K_3) \oplus K_4)$ at time $T - t_4$.
3. At time T, the prover at position P computes messages $K_{i+1} = \text{PRG}(X_i, K_i) \oplus K_{i+1}'$ for $1 \leq i \leq 3$. The prover returns K_4 to all verifiers.
4. All verifiers check that the string K_4 is received at time $(T + t_i)$ and that it equals the K_4 that they pre-picked. If these verifications succeed, the position claim of the prover is accepted and it is assumed to be indeed at position P. Otherwise, the position claim is rejected.

The completeness of this protocol is as follows. Note that since the verifiers picked K_4 before the execution of the protocol, they can verify the response of a prover without storing any of the large random strings. To informally argue security of the protocol, note that in this protocol, instead of using the output of the PRG as an input key in the next round, one treats the output as one secret share of the key to be used. The other share of this key is broadcast in the clear. Now, if one of the shares of an additive secret sharing scheme is random, then the secret is hidden. Hence, by the security of the protocol in which verifiers could store the large random strings, it follows that this protocol is also secure.

6 Computational Position based Key Exchange

Informally, position based key exchange should have the property that if there is a prover at the claimed position P, then at the end of the protocol, the verifiers should share a uniform key K with it while for a group of colluding adversaries (none of whom is at P) K should look indistinguishable from a key drawn uniformly at random. This also implies that in the absence of a prover at position P, such a group of adversaries should be unable to execute the key exchange protocol on their own to obtain a shared key with the verifiers. In this section, we show how to compile any 1-round *information theoretically* secure positioning protocol SP in our bounded retrieval model along with any unauthenticated key-exchange protocol KE to obtain an authenticated computational position based key exchange protocol CKE in the BRM.

Any time of response based 1-round secure positioning protocol SP in the BRM begins with (possible) messages from verifiers to the prover at P and ends with a response from the prover at P to all or some of the verifiers (that needs to reach the verifiers within some particular time). For simplicity, we first show how to construct computational position based key exchange with the assumption that the prover has

the (authentic) public key of the verifiers before the protocol starts. We then show how to remove this assumption by using techniques based on non-malleable commitments ([DDN00, PR08]).

Let the verifiers generate a public key - secret key pair (pk, sk) of a CCA2 secure public key encryption scheme [RS91, DDN00] (i.e., a scheme secure against adaptive chosen ciphertext attacks). We assume that any prover would have access to pk. Let $\mathsf{E}(pk, m)$ denote an encryption of a message m with public key pk and let $\mathsf{D}(sk, c)$ denote the decryption of ciphertext c with secret key sk. Let KE be an unauthenticated key exchange protocol. Our protocol for computational position based key exchange CKE is as follows:

1. First, the prover and an arbitrary verifier V_j carry out the unauthenticated key exchange protocol KE to obtain shared key k. Let the entire transcript of this protocol be denoted by T_k. At the end of this protocol, verifier V_j sends T_k to all other verifiers through the private channel.

2. The verifiers then carry out the first message of the secure positioning protocol SP with the prover at P. Let the response which the prover needs to broadcast to verifier V_i be K_i for all i.

3. For all i, the prover at point P computes $c_i = \mathsf{E}(pk, T_k||K_i)$ and broadcasts it to V_i.

4. Verifier V_i computes $\mathsf{D}(sk, c_i)$ and parses the obtained plaintext as $T'_k||K'_i$. If the response K'_i for all verifiers completes the secure positioning protocol SP successfully, then all the verifiers check if $T'_k = T_k$, and if so accept k as the shared secret key. Otherwise, the verifiers reject the response.

The high level intuition behind the security of this protocol is as follows. First, note that the above protocol CKE is also an information theoretically secure positioning protocol. This is because if an adversary can respond with $\mathsf{E}(pk, T_k||K_i)$ to V_i in CKE, then there exists a (computationally unbounded) simulator that can respond with K_i to V_i in SP. Now, by the security of SP, it follows that for an adversary to be successful in the CKE protocol, there must also be an honest prover at P sending out an encrypted message in the last round. Note that by the security of KE, it follows that an adversary which is passive during the execution of KE between the prover and the verifiers does not obtain any information about key k. Now, an *active* adversary in KE must modify the transcript T_k (as observed by the prover) to T'_k. This means that the adversary must also maul the encryption c_i to succeed in the protocol and this breaks the CCA2 security of E.

In order to remove the assumption that verifiers authentically know the public key of the verifiers, we extend the above idea in the following way. The verifiers and the device at P carry out the first two steps as in the above protocol. As the response, the device at P sends $\mathsf{C}(T_k||K_i; r)$ where $\mathsf{C}(m; r)$ denotes a perfectly binding non-interactive commitment to message m using randomness r. Next the device, using a non-malleable zero knowledge proof of knowledge protocol, proves to the respective verifiers that it knows an opening to this commitment. Finally, the prover sends openings to the commitments to the respective verifiers. The verifiers check the openings and all timings and accept the key k if all these checks succeed. We refer the reader to Appendix F for a description of the protocol and a formal proof of security.

7 Information theoretic Position based Key-Exchange

In this section, we present an information theoretic protocol to achieve position based key exchange. The overview of our protocol can be found in Figure 2. We start with some intuition behind our protocol and

the techniques required to prove its security. Let us first consider the case of one dimension. We extend the protocol for secure positioning in one dimension presented earlier for the case of key exchange as follows. Instead of only one verifier V_2 sending a "large" string (drawn from a reverse block entropy source), both the verifiers send one large string each. More precisely, the verifier V_1 sends a key K_1 and a large string X_2 while the verifier V_2 sends a large string X_1 such that all of them meet at the claimed position P at the same instance of time T. The computation of the final key K_3 is done by the prover as follows: set $K_2 = \mathrm{PRG}(X_1, K_1)$, $K_3 = \mathrm{PRG}(X_2, K_2)$.

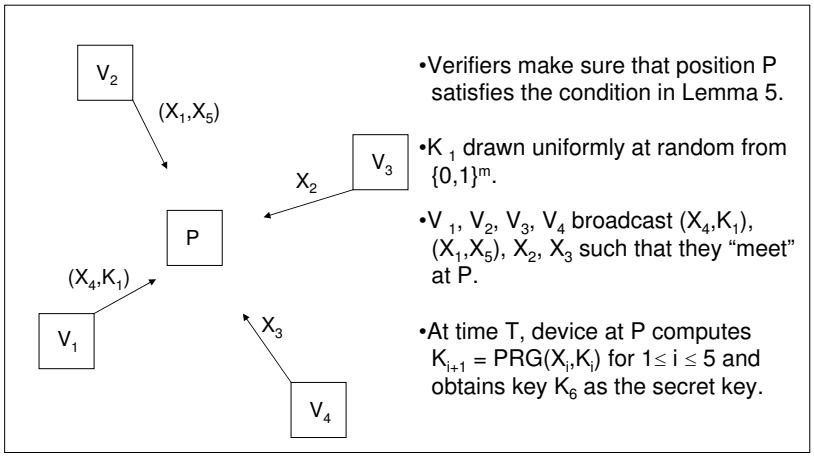

Figure 2: Position based key exchange in 3-Dimensions

To see the intuition behind why this protocol is a secure one dimensional information theoretic position based key exchange, let us consider the state of the system at time T. Adversaries between V_1 and P (say, adversaries of type I) have stored $(K_1, A(X_2, K_1))$ while adversaries between P and V_2 (say, adversaries of type II) have stored $A(X_1)$. After time T, the adversaries of type I can compute K_2 thus transitioning their state to $(K_2, A(X_2, K_1))$ while adversaries of type II can only transition their state to $A(X_1), K_1, A(X_2, K_1)$. Thus it seems that to both these types of adversaries together, the final key K_3 remains uniform. Indeed it turns out that this intuition is sound and the above is a secure one dimensional information theoretic position based key exchange protocol.

For three dimensions, we have the prover to be inside the tetrahedron defined by the four verifiers. Now, one can similarly try to extend the three-dimensional information theoretic secure positioning protocol presented earlier to achieve three-dimensional information theoretic position based key exchange. Simply add a fourth long string X_4 to be sent by V_1 in the natural way. However, it turns out that the above idea is not sufficient because of the fact that there might be adversaries (far) outside this tetrahedron trying to compute the key exchanged between the verifiers and an honest prover. In the case of secure positioning, such adversaries would be too late in sending their response to the verifiers (there is no honest prover to aid these adversaries). However, the key exchange scenario requires that once the verifiers and the honest prover get a shared key after running the protocol, this key should be uniform to the adversaries even at a much later point in time.

In contrast to what the intuition might suggest, the first problem we face is that there are certain regions in the tetrahedron defined by the verifiers such that if the claimed position P lies within one of these regions, there exists points, other than the point P, in the three dimensional space (but outside the

tetrahedron) where the messages broadcast by the four verifiers all meet simultaneously. Thus, if there is an adversary located at such a point, it can compute the final key shared between the verifiers and the honest prover simply by following the algorithm of the honest prover. To overcome this problem, we characterize such regions of the tetrahedron (see Lemma 5; in Appendix H we further show that the remaining region is a still a large fraction of the tetrahedron) and exclude them from the area from which position claims are accepted. That is, given an area from which position claims need to be accepted, Lemma 5 depicts the acceptable positioning of the verifiers so that they can verify the position claims from that area.

The second main problem that arises is that even if the messages broadcast by the verifiers do not all meet at a single point (other than P), there of course could be multiple colluding adversaries which utilize different information available at multiple different points at different time instances to try to compute the final key. Indeed, it can be shown that there is in fact an explicit attack on the protocol discussed earlier (that is, the protocol resulting from a natural extension of our three-dimensional secure positioning protocol where the verifiers broadcast four long strings) which allows multiple colluding adversaries to completely recover the key exchanged between the verifiers and an honest prover. To solve the above problem, we introduce a fifth long string in a similar way as before. Introducing this fifth long string allows us to construct a geometric argument, along with a reduction argument relying on techniques from [DP07], that multiple colluding adversaries do not have sufficient information, and our security proofs go through. Our final protocol is given in Figure 2. Our security proofs are a careful combination of the following two components:

- A geometric argument which rules out a "nice" way for adversaries to recover the final key exchanged. In other words, very roughly, there does not exist a strategy for multiple colluding adversaries to perform the operation $K_{i+1} = \text{PRG}_i(X_i, K_i)$ in sequence for each $i \in [5]$ to recover the final key K_6.
- A reduction argument relying on the techniques from [DP07] to prove the final security of our protocol. In more detail, given the above geometric argument, if there exists an adversarial strategy that can distinguish the final key K_6 from uniform in our protocol, then we can construct an adversarial strategy to contradict the security guarantees of an intrusion resilient secret sharing scheme (as defined and constructed in [DP07]).

Complete details of our protocol and the security proofs are given Appendix G. The completeness of the above protocol described relies on the assumption that the verifiers can store the long strings they generated to be able to compute the final key K_6 themselves. In the appendix, we show that, as with the case of secure positioning, this assumption can be relaxed by using the same secret sharing technique introduced in Section 5.

8 Applications: Constructing Various Position based Cryptosystems

In this section, we consider the case where there are several devices denoted by $\{P_1, ..., P_n\}$. The devices are assisted by a trusted infrastructure (i.e., a set of verifiers) and are trying to perform a position based task amongst themselves. We show that using position-based key exchange as a fundamental protocol, a number of protocols for realizing more complex tasks can be constructed in a straightforward manner. We

discuss how to (a) perform position based Multi-party computation, and, (b) setup position based Public Key Infrastructure.

Position based Multi-party Computation Consider a set of parties at different geographic locations. Within themselves, they wish to verify each other's position and run multi-party computation with inputs possibly dependent upon their positions. For example, a set of parties might wish to see which party is located at a place having the best weather conditions. The parties are willing to trust each other with the input supplied (i.e., the weather condition) only if it is clear that each is physically present at the place whose weather is being supplied.

We assume that every party P_i shares a secret key K_i with the trusted infrastructure (this is done through position-based key exchange). For all pairs of parties $(P_i, P_j), j \neq i$, the infrastructure generates a shared secret key $K_{i,j}$ and broadcasts $E_{K_i}(K_{i,j})$ and $E_{K_j}(K_{i,j})$, where $E_K(M)$ denotes an encryption of M under key K. Thus, P_i and P_j obtain a shared secret key $K_{i,j}$ and hence a secure channel between them. Once every pair of parties have a secure and private communication channel, standard generic multi-party computation protocols ([BoGW88],[GMW87]) can be run to achieve position based multi-party computation. This enables us to realize virtually every functionality securely.

Establishing a Position-based Public Key Infrastructure In a regular public key infrastructure (PKI), each party typically has a certificate from a trusted CA. The certificates provide a binding between the public key and the identities. In other words, a certificate gives a guarantee about the *identity* of the owner of the public key in question (thus saying that it is safe to use a particular public key to encrypt email for a recipient).

In a position based PKI, the positioning infrastructure assign a certificate to each party binding its position with a public key. The certificates would provide a guarantee that the public key indeed belongs to a party at position P (rather than a party with an identity I). Hence the certificate (along with the key pair) could be used for public key encryption and signature based applications where the geographic positions are valuable. The certificate could also have an expiration time based on the application (depending upon how often the parties are likely to move).

Acknowledgments. We thank Yevgeniy Dodis for interesting discussions. We also thank Harish Rajagopalan for the simulations that helped identify the regions within the tetrahedron where information theoretic position-based key exchange is possible.

References

[ADR02] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In *CRYPTO*, pages 65–79, 1999.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.

[BC94] Stefan Brands and David Chaum. Distance-bounding protocols. In *EUROCRYPT '93: Advances in cryptology*, pages 344–359. Springer-Verlag New York, Inc., 1994.

[BoGW88] Michael Ben-or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC '88: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 2-4 May, Chicago, Illinois, USA*, pages 1–10, 1988.

[Bus04] Laurent Bussard. *Trust Establishment Protocols for Communicating Devices*. PhD thesis, Eurecom-ENST, 2004.

[CCM98] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *FOCS*, pages 493–502, 1998.

[CCS06] Srdjan Capkun, Mario Cagalj, and Mani Srivastava. Secure localization with hidden and mobile base stations. In *IEEE INFOCOM*, 2006.

[CDD+07] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2007.

[CH05] Srdjan Capkun and Jean-Pierre Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE INFOCOM*, pages 1917–1928, 2005.

[CHH09] Jerry T. Chiang, Jason J. Haas, and Yih-Chun Hu. Secure and precise location verification using distance bounding and simultaneous multilateration. In David A. Basin, Srdjan Capkun, and Wenke Lee, editors, *WISEC*, pages 181–192. ACM, 2009.

[CLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In *CRYPTO '97: 17th Annual International Cryptology Conference, Advances in Cryptology*, pages 292–306, 1997.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DHRS04] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *TCC*, pages 446–472, 2004.

[Din01] Yan Zong Ding. Oblivious transfer in the bounded storage model. In *CRYPTO*, pages 155–170, 2001.

[Din05] Yan Zong Ding. Error correction in the bounded storage model. In *TCC*, pages 578–599, 2005.

[DM04a] Stefan Dziembowski and Ueli M. Maurer. On generating the initial key in the bounded-storage model. In *EUROCRYPT*, pages 126–137, 2004.

[DM04b] Stefan Dziembowski and Ueli M. Maurer. Optimal randomizer efficiency in the bounded-storage model. *J. Cryptology*, 17(1):5–26, 2004.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS '07: Proceedings of the 48th Annual IEEE Foundations of Computer Science*, 2007.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS '08: Proceedings of the 49th Annual IEEE Foundations of Computer Science*, 2008.

[DR02] Yan Zong Ding and Michael O. Rabin. Hyper-encryption and everlasting security. In *STACS*, pages 1–26, 2002.

[Dzi06a] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.

[Dzi06b] Stefan Dziembowski. On forward-secure storage. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 251–270. Springer, 2006.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Lu04] Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *J. Cryptology*, 17(1):27–42, 2004.

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptology*, 5(1):53–66, 1992.

[MSTS04] Tal Moran, Ronen Shaltiel, and Amnon Ta-Shma. Non-interactive timestamping in the bounded storage model. In *CRYPTO*, pages 460–476, 2004.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[PR08] Rafael Pass and Alon Rosen. New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.*, 38(2):702–752, 2008.

[RRV02] Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO 1991*, pages 433–444, 1991.

[SP05] Dave Singelee and Bart Preneel. Location verification using secure distance bounding protocols. In *IEEE Conference on Mobile Adhoc and Sensor Systems Conference*, 2005.

[SSW03] Naveen Sastry, Umesh Shankar, and David Wagner. Secure verification of location claims. In *WiSe '03: Proceedings of the 2003 ACM workshop on Wireless security*, pages 1–10, 2003.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

[Vad04] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.

[VN04] Adnan Vora and Mikhail Nesterenko. Secure location verification using radio broadcast. In *OPODIS '04: 8th International Conference on Principles of Distributed Systems*, pages 369–383, 2004.

[WF03] Brent Waters and Edward Felton. Secure, private proofs of location. Technical report, Department of Computer Science, Princeton University, 2003.

[ZLFW06] Yanchao Zhang, Wei Liu, Yuguang Fang, and Dapeng Wu. Secure localization and authentication in ultra-wideband sensor networks. *IEEE Journal on Selected Areas in Communications*, 24:829–835, 2006.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Struct. Algorithms*, 11(4):345–367, 1997.

Appendix

A Related Work

Secure Positioning. The problem of position-based cryptography as such has not been studied before. However, secure positioning is a well-studied problem in the field of wireless security. There have been several proposed protocols ([SSW03, SP05, Bus04, CH05]). We note that all these protocols are susceptible to an attack in which an adversary clones a device (along with any keys and authentication mechanisms that it may have) and uses it in a controlled manner at various locations (other than the location to be proved) to falsely prove to the verifiers that it is at the designated position.

One of the earlier techniques used for secure positioning has been the distance bounding protocol [BC94] that uses time-of-response as a parameter to verify the location of a prover. In this protocol, the prover and verifier start with random numbers a and b respectively. They exchange the random numbers bit by bit rapidly. At the end of this exchange, the verifier measures the time taken for the protocol. Assuming that the bits travel at the speed of radio waves, the verifier's location can then be verified. Several improvements of this protocol (with respect to different adversarial models) have been discussed [SSW03, VN04, Bus04]. These protocols either consider the model where an adversary, in between the prover and the verifier, plays man-in-the-middle or a model where a single adversarial prover tries to prove to the verifier that it is at a false location.

Capkun et al. [CH05] give a protocol for secure localization (positioning) that is secure under a single adversary who tries to falsely prove to the verifiers that he is at the specified location. The protocol works in d-dimensional space. It is insecure under adversaries who can clone devices. The protocol in [SP05] is also susceptible to multiple adversaries (having cloned devices) attacking the system. Further, it assumes that the prover and verifiers share cryptographic keys in advance. [ZLFW06] give a protocol for a prover to prove its location in the presence of adversaries. These protocols are also not secure in the presence of multiple adversaries with cloned devices. The protocol in [WF03] makes use of cryptographic setup assumptions and is also susceptible to multiple adversaries (with cloned devices) attack.

In [CCS06], they consider a model that makes use of covert base stations. These are base stations whose position is not known to the prover and to the adversaries. Based on this, they provide solutions to secure positioning with hidden and mobile base stations. The protocol in [CCS06] is also susceptible to multiple colluding adversaries, although the attack required is more subtle than in other cases. We outline the attack in Section C.

The Bounded Storage Model. The bounded storage model for private-key cryptography was introduced by Maurer [Mau92]. The efficiency and the security of the private-key cryptosystems in the model were improved in a sequence of work [CM97, AR99, ADR02, DR02, DM04b, Lu04, Vad04]. Notably in [ADR02, DR02], it is shown that protocols in the model can have the novel property of "everlasting security". That is, the security is maintained even if after the protocol is used, the key is revealed to the adversary and the adversaries computational and storage capacity becomes unbounded.

The works of [DM04b, Lu04, Vad04] give extremely efficient cryptosystems for this model, with [Vad04] having nearly optimal key length and storage requirements. The work of [Vad04] introduces the "sample then extract" paradigm, that is particularly relevant to our work. The idea of the "sample then extract"

paradigm is to use a sampler to select bits from a random source and then use the extractor to convert these bits to a nearly uniform distribution. This immediately gives rise to locally computable extractors which [Lu04] shows are very useful for creating cryptosystems in the bounded storage model.

The bounded retrieval model (BRM) was introduced and studied in various related contexts by Crescenzo et al. [CLW06] and Dziembowski [Dzi06a, Dzi06b]. Dziembowski [Dzi06a] initiated the study of *intrusion resilient* key exchange in this model and proposed constructions in the random oracle model. Cash et al. [CDD$^+$07] subsequently gave a construction in the standard model using a UC-secure password authenticated key exchange. Other problems studied in this model include password based authentication [CLW06] and forward secure storage [Dzi06b]. Recently, Dziembowski and Pietrzak [DP07] introduced *intrusion resilient secret sharing* where shares of a secret (stored on different machines) are made artificially large so that it is hard for an adversary to retrieve a share completely even if it breaks into a storage machine. However the reconstruction of the secret can be done by the parties storing the shares very efficiently by running a protocol. Dziembowski and Pietrzak [DP08] elegantly also use the bounded retrieval model to obtain provable security of stream ciphers against side-channel attacks.

We note that in the current work, we use the work of [DP07] on Intrusion Resilient Secret Sharing schemes as a starting point. We build on these techniques and combine them with geometric arguments to prove the security of our protocol. Furthermore, we also prove that in the protocol of [DP07], even when multiple adversaries may break into the storage machine in parallel and then collude, the secret will be statistically close to a uniformly random secret.

B The Model

In this section, we discuss our setting in further detail. All the parties are modeled as Interactive Turing Machines (ITMs). There are three categories of ITMs: Prover, Verifier and Adversary. Apart from these ITMs, to model the environment (and the communication channel), we also assume the existence of a (hypothetical) trusted ITM P_{env}. All ITMs have local clocks. We assume that the clocks of all verifiers are synchronized (we stress that this assumption is only for clarify of exposition and can be relaxed). We do not require clock synchronization between verifiers and the honest prover, but we require that the pace of their clocks be the same. The adversaries may (or may not) synchronize their clocks and have them run with a different pace. The trusted ITM P_{env} can read all local clocks and also maintain its own clock (which is the global time reference). We treat time and space as continuous (rather than discrete). We assume that messages travel at a speed equal to that of radio waves (which is the same as the speed of light).

In the beginning, P_{env} gives as input to each machine (prover, verifiers and adversaries) its own position (as a point in the d-dimensional space) and the security parameter κ. P_{env} also gives every machine (including the verifiers themselves), the positions and unique identifiers of all the verifiers. The verifiers and the adversaries are given the claimed position of the prover. P_{env} can read output tapes and write messages on input tapes of all machines at the appropriate time as discussed later.

A *Verifier* ITM can send the following types of messages:

1. **Broadcast messages:** A broadcast message originating at a position P, travels in concentric hyperspheres centered at P. It travels with equal speed in all directions and reaches a position P'

after time t from the moment of broadcast, where t is the time needed by radio waves to travel the distance between P and P'.

2. **Directional messages:** A directional message originating at a position P, travels in a region (specified by its angle with respect to the base axes) of concentric hyperspheres centered at P. The region of the hypersphere can be specified by the angles with respect to all the base axes in d-dimensional space (for example, in the 2-dimensional case, the Verifier selects a sector of a circle). The message travels within this region and it reaches position P' (that is in this region) after time t from the moment of broadcast, where t is the time needed by radio waves to travel the distance between P and P'. The idea behind permitting such messages is that they can be readily sent using a directional antenna.

3. **Private multi-cast messages:** We allow a verifier to talk to other verifiers via a private channel. A private multicast message is used by a verifier V to send a message to a subset of other verifiers such that no adversary or honest party gains any information about this message. This can be achieved, for example, by having a separate (wired) channel shared by the verifiers, or, by sharing a secret key among the verifiers which can be used to encrypt messages.

An *Adversary* ITM can send the following types of messages:

1. **Broadcast messages:** This type of message can be used by an adversary to send messages to all other Turing machines as described earlier.

2. **Directional messages:** This type of message can be used by an adversary to send messages in a particular region of a hypersphere as described earlier. An adversary can send such a message by using a directional antenna.

3. **Private multi-cast messages:** We assume that adversaries have a covert channel through which they can communication without the knowledge of the verifiers. This can be done, for example, by broadcasting messages at a frequency not monitored by the verifiers, or, by encrypting the data with a shared secret key.

A *Prover* ITM can send the following types of messages:

1. **Broadcast messages:** This type of message can be used by a prover to send messages to all other Turing machines as described earlier.

2. **Directional messages:** This type of message can be used by a prover to send messages in a particular region of a hypersphere a described earlier.

To send a message, an ITM writes it to its output tape (along with the message type and any parameters needed to fully specify its direction/recipients). The trusted ITM P_{env} instantaneously reads it and runs a mental experiment (to check the message trajectory) in order to be able to compute the time when other ITMs should receive it. P_{env} writes the message to the input tape of the receiving ITMs at the appropriate time. Along with the message, if the sending ITM is a verifier, P_{env} also writes its identifier.

The default message sent by any ITM is a broadcast message. If the message is of any other type, we would specify its type explicitly. We now define secure positioning and position based key exchange.

Secure Positioning. A positioning protocol $SP(Ver, Prov, d)$ in d-dimensional space is a set of ITMs $Ver = \{V_1, V_2, ..., V_n\}$ at positions $pos_1, pos_2, ..., pos_n$ respectively, that take as input a claimed position P' of a prover $Prov$ at position P, and jointly return `"Accept"` after interacting with the honest prover $Prov$ in the absence of any adversarial parties, if $P' = P$.

A positioning protocol is said to be *secure*, if for all sets of adversary ITMs $\{A_1, A_2, ..., A_k\}$ at positions $apos_1, apos_2, ..., apos_k$, and for all positions P in the tetrahedron enclosed by $pos_1, pos_2, ..., pos_n$, with $P \neq apos_i, \forall\ i$, the set of ITMs $\{V_1, V_2, ..., V_n\}$ at positions $pos_1, pos_2, ..., pos_n$, jointly return `"Accept"` with probability ϵ, where ϵ is negligible in the security parameter κ.

Position based Key Exchange. A position-based key exchange protocol in d-dimensional space, $KE(Ver, Prov, d)$ is a set of ITMs $Ver = \{V_1, V_2, ..., V_n\}$ at positions $pos_1, pos_2, ..., pos_n$ respectively, that take as input a claimed position P', of a prover $Prov$ at position P. After interacting with the honest prover $Prov$ in the absence of any adversarial parties, the prover and the verifiers (jointly) should output a key K.

Let $K = KE(Prov, Ver, d)$. Let the verifiers output a secret key K after the protocol interaction (with adversaries and/or possibly the honest prover $Prov$). The key exchange protocol KE is *secure* if for all sets of adversary ITMs $Adv = \{A_1, A_2, ..., A_k\}$ at positions $apos_1, apos_2, ..., apos_k$ and for all positions P in the tetrahedron enclosed by $pos_1, pos_2, ..., pos_n$, with $P \neq apos_i, \forall\ i$,

$$|\Pr[Adv(K) = 1] - \Pr[Adv(U) = 1]| < \epsilon$$

where U is a uniform string of appropriate length and ϵ is negligible in the security parameter κ.

As is clear from the above definitions, for the task of secure positioning, either we have an (honest) prover at the claimed position, or we have a set of adversaries none of whom is at the claimed position. The adversaries take part in a positioning protocol in an attempt to falsely prove the claimed position. Hence, we do not consider the scenario where there is an honest prover at the claimed position *as well as* a set of adversaries (since in this case, the adversaries could just do nothing and the verifiers will output `Accept`). However, in the case of key exchange, we consider adversaries *along* with the honest prover as well. This is because, in this case, we are concerned with preserving the key privacy of the prover and verifier and this might be compromised by an adversary that can also observe the messages sent between the prover and the verifier.

The above is our so called Vanilla Model where we prove the impossibility of realizing the most basic position based task (i.e., secure positioning). However as noted earlier, all our positive results are in the bounded retrieval model. More details about that follow.

Our bounded retrieval model is the same as the Vanilla model except for the following changes:

- Verifiers "possess" a reverse block entropy source (defined in Section D) capable of generating strings with high min-entropy, say $(\delta + \beta)n$, where n is the length of the string (and $0 < \delta + \beta < 1$, it is also called min-entropy rate). By possessing a reverse block entropy source, we mean that either the verifier itself is capable of generating such a string of high min-entropy, or it has a randomness source (located at the same point in space as itself) which generates and broadcasts such string. We do not assume that the verifiers can retrieve and store the broadcasted string of data themselves.

- There exists a bound βn on the total amount of information the adversaries can store as the information passes at a high speed. Adversaries can store arbitrary functions of the information provided the length of the output of the function is $\leq \beta n$. The retrieval bound βn could be any constant fraction of the min-entropy $(\delta + \beta)n$. The honest parties (including the verifiers) are required to have a storage capacity of only $O(\kappa \cdot \log(n))$.

- Let X be a string having large min-entropy as before. The message passing for X is now modeled as follows. The trusted ITM P_{env} keeps track of the amount of information each ITM has retrieved so far. The sender (which is a verifier) generating X writes it directly on the input tape of P_{env} (who reads it instantaneously). As before, P_{env} runs a mental experiment to compute the time when other ITMs should receive it.

  Let t be a time instance when an ITM A is supposed to be receiving the strings $X_1, \cdots, X_m$ (an ITM may receive multiple strings at the same time). The ITM A retrieves information from the received strings in several iteration. In each iteration, A specifies an index i and a function f to P_{env}. The trusted ITM P_{env} computes $f(X_i)$ and writes it back to the input tape of A as long as the retrieval bounds are not violated[1]. Otherwise P_{env} truncates the output $f(X_i)$ to the appropriate size (possibly zero bits) before writing it to the input tape of A. The ITM A is allowed to run as many iterations as it wants. Given a string X, we stress that every ITM (including the verifier sending X) has to follow this procedure to retrieve any information about X. Observe that we do not allow the ITM A to directly retrieve a joint arbitrary function of the strings $X_1, \cdots, X_m$.

Relaxing Assumptions. For clarity of exposition during our positive results, we make the following assumptions. We assume that the devices can read bits from the string and perform lightweight computations instantaneously. It is easy to take a protocol with these assumptions and "compile" it into another protocol (which allows time for computations). Consequently, the difference between the allowed time (for honest devices) and the actual time taken by an adversary for an operation will determine the *precision* with which positioning can be achieved. The same technique can also be used to relax the assumption that the honest prover is located exactly at the same point in space as claimed (and thus allow a small variation in the claimed and the actual positions of the honest prover). The basic idea of this compiler is as follows.

Without loss of generality, the verifiers wish to allow for a fixed amount of time (say t) for computation *immediately* after the honest prover receives a message m. Then after the instance at which the prover is supposed to receive m (say T), the verifiers and the prover come to a "standstill" for time t (while the prover is doing the required computation). In other words, all the honest parties (prover and the verifier) delay sending and receiving rest of the messages of the protocol by time t. Thus, any message that was supposed to be sent or received at time $T + t_1$ in the original protocol, will now be sent or received at $T + t_1 + t$. This yields a new protocol in which a computation time t is allowed at a specific point. By applying this compiler repeatedly, we can achieve a protocol to allow computation time at multiple points in time.

[1]For example, if A is an adversary, its information retrieval length will be added to the total information retrieved by all adversaries and if the adversarial information retrieval bound is not violated, $f(X_i)$ will be written onto the input tape of A.

C Extensions of the lower bound in the Vanilla Model

In Section 3, the generic attack works as long as every adversary knows the source of every message. In other words, all adversaries know which one of the verifiers sent a particular message. Below, we sketch how this restriction can also be removed. In addition to the set of adversaries as discussed earlier, there exists a set of n adversaries $\{B_1, B_2, ..., B_n\}$, such that B_i is at a distance ϵ_i close to V_i, where ϵ_i is such that B_i is the first adversary to receive any message sent by V_i to P. Now, every adversary B_i upon seeing a message without a label, appends a label to the message. In particular, when V_i sends out a message, adversary B_i will be the first to receive this (as it is ϵ_i close to V_i). B_i will append a label V_i along with this message and then send the message to other adversaries. This message can be sent using private multicast message (or in other words, over the covert channel) so that no verifier knows that the adversaries are appending labels to messages. Now, the attack works as before and the set of l adversaries $\{A_1, ...A_l\}$ can run the protocol discussed earlier. This proves that *Secure Positioning* in the Vanilla model (even with this relaxation) is impossible to achieve.

C.1 Attacks on [CCS06].

The authors [CCS06] present protocols for secure positioning based on covert base stations (i.e., hidden verifiers) and based on mobile covert base stations (i.e., hidden and moving verifiers). We give a high level sketch of our attacks on these protocols. We show the attack in the two dimensional case (on a disk of radius R as in [CCS06]), but these results can be extended to the more general cases. A very high level description is given below. Please refer Appendix I for more details.

Stationary Covert Base Stations For the stationary base station case, we assume there is a single covert base station, but the attack can be easily generalized to multiple covert base station. We place three devices in the positioning region such that they are not in a line. Each device will be able to locate a sector extending out from it containing the covert base station as follows. The devise will broadcast legitime positioning interactions messages, however it will only do so on certain sectors extending out from it using a directional antenna. Thus when a positioning interaction is rejected the device will know that a covert base station is where he did not broadcast the message. The device will be able to conduct a binary search on the area and find a thin sector having the covert base station. The three devices will then combine their information and will be able to bound the area in which the covert base station could be present.

Mobile Covert Base Stations In this case, we assume that the covert mobile base station will not come within some radius r of the position. If this is not the case then the mobile base station can simply physically check that the device is in the location. Thus this condition is necessary for this model to be meaningful.

For this attack, we employ multiple adversaries that are on the circle of radius r around the position which they are trying to fake. The adversaries will broadcast with the appropriate time delay outward in a thin enough sector such that for the entire sector, the signal will reach the covert base station at the appropriate time. If we employ enough adversaries on the circle, we can show through a geometric

argument that for any constant number of randomly moving covert base stations as in [CCS06], this attack will have a constant probability of success.

D Bounded Storage Model Primitives

In this section, we introduce a new bounded storage model primitive called a BSM Entropy Generator (EG). We describe its generic construction using averaging samplers. We also briefly recall BSM pseudorandom generators introduced by [Vad04]. Finally, we describe the instantiation of each of these using fast samplers and extractors suitable for our purposes.

D.1 Preliminaries

The statistical difference between two random variables X, Y taking values in a universe $\mathbb{U}$ is defined to be

$$\Delta(X,Y) \stackrel{def}{=} \max_{S\subset\mathbb{U}} \Big|\Pr[X\in S] - \Pr[Y\in S]\Big| = \frac{1}{2}\sum_{x\in\mathbb{U}} \Big|\Pr[X=x] - \Pr[Y=x]\Big|$$

We say X and Y are ε-close if $\Delta(X,Y)\le\varepsilon$.

We define *min-entropy* of a random variable X as $H_\infty(X) \stackrel{def}{=} \min_x \log(1/\Pr[X=x])$. X is called a *k-source* if $H_\infty(X)\ge k$, i.e., for all x, $\Pr[X=x]\le 2^{-k}$.

We also define reverse block entropy as follows.

Definition 1 ([Vad04]) *A sequence of random variables $X_1, X_2, \cdots$, each distributed over $\{0,1\}^n$ is a reverse block source of min entropy rate α if for every $t\in\mathbb{N}$ and every $x_{t+1}, x_{t+2}, \cdots$, the random variable $X_t|_{X_{t+1}=x_{t+1}, X_{t+2}=x_{t+2},\cdots}$ is an αn-source.*

Next, we define a few objects which will be helpful in the construction of our bounded storage model (BSM) primitives later on.

Definition 2 ([NZ96]) Ext*: $\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$ is a strong (k,ε)-extractor if for every k-source X, the distribution $K\times\mathrm{Ext}(X,K)$ where $K \stackrel{R}{\leftarrow} U_d$ is ε-close to $U_d\times U_m$.*

Definition 3 ([Vad04]) *A function* Samp*: $\{0,1\}^m\to[n]^\ell$ is a (μ,θ,γ)-*averaging sampler *if for every function $f:[n]\to[0,1]$ with average value $\frac{1}{n}\sum_i f(i)\ge\mu$, it holds that*

$$\Pr_{(i_1,\cdots,i_\ell)\stackrel{R}{\leftarrow}\mathrm{Samp}(U_m)}\left[\frac{1}{\ell}\sum_{j=1}^{\ell} f(i_j) < \mu-\theta\right] \le \gamma.$$

Samp *has distinct samples if for every $x\in\{0,1\}^m$, the samples produced by* Samp(x) *are all distinct.*

Definition 4 ([HILL99]) *Let $h:\{0,1\}^{l_n}\times\{0,1\}^n\to\{0,1\}^{m_n}$ be a* **P***-time function ensemble. For a fixed $y\in\{0,1\}^{l_n}$, we view y as describing a function $h_y()$ that maps n bits to m_n bits. Then, h is a universal hash function if, for all $x\in\{0,1\}^n$, $x'\in\{0,1\}^n\backslash\{x\}$, for all $a,a'\in\{0,1\}^{m_n}$,$K \stackrel{R}{\leftarrow} U_{l_n}$*

$$Pr[(h_K(x)=a\cap(h_K(x')=a')] = 1/2^{2m_n}.$$

D.2 Primitives and Constructions

In this section we will describe the two primitives we will be using in the rest of the paper, namely BSM entropy generator (EG) and BSM pseudorandom generator (PRG). We also show how to construct them using samplers and strong extractors.

D.2.1 BSM Pseudorandom Generators

Vadhan [Vad04] introduced BSM pseudorandom generators and discussed their construction using locally computable extractors. We briefly recall his results in this section.

Definition 5 ([Vad04]) PRG*:* $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is an* ε*-secure BSM pseudorandom generator for storage rate* β *and min-entropy rate* α *if and only if for every* αn*-source* X *on* $\{0,1\}^n$ *and for every function* A*:* $\{0,1\}^n \to \{0,1\}^{\beta n}$*, the random variable* $(\mathrm{PRG}(X,K), A(X), K)$ *is* ε*-close to* $(U_m, A(X), K)$*, where* $K \stackrel{R}{\leftarrow} \{0,1\}^d$*.*

Theorem 3 ([Vad04], also implicit in [Lu04]) *If* Ext*:* $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a strong* $(\delta n - \log(1/\varepsilon), \varepsilon)$*- extractor, then for every* $\beta > 0$*,* PRG $=$ Ext *is a* 2ε*-secure BSM pseudorandom generator for storage rate* β *and min-entropy rate* $\beta + \delta$*.*

Although the above theorem suggests that BSM PRGs can be constructed using any strong extractor, it is desirable to minimize the number of bits read[2] from the source X. To achieve this goal, Vadhan introduced locally computable extractors.

Definition 6 ([Vad04]) Ext*:* $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is* t*-locally computable (or* t*-local) if for every* $r \in \{0,1\}^d$*,* $\mathrm{Ext}(x,r)$ *depends on only* t *bits of* x*, where the bit locations are determined by* r*.*

The use of locally computable extractors in Theorem 3 yields efficient BSM PRGs reading only a small number of bits from the source. Vadhan shows how to generically construct t-local extractors given a regular extractor and an averaging sampler.

Theorem 4 ([Vad04]) *Let* $1 \geq \delta \geq 3\tau > 0$*. Suppose that* Samp*:* $\{0,1\}^r \to [n]^t$ *is a* (μ, θ, γ)*-averaging sampler with distinct samples for* $\mu = (\delta - 2\tau)/\log(1/\tau)$ *and* $\theta = \tau/\log(1/\tau)$ *and that* Ext:$\{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ *is a strong* $((\delta - 3\tau)t, \varepsilon)$ *extractor. Define* Ext′:$\{0,1\}^n \times \{0,1\}^{r+d} \to \{0,1\}^m$ *by*

$$\mathrm{Ext}'(x,(y_1,y_2)) = \mathrm{Ext}(x_{\mathrm{Samp}(y_1)}, y_2).$$

Then Ext′ *is a* t*-local strong* $(\delta n, \varepsilon + \gamma + 2^{-\Omega(\tau n)})$ *extractor.*

D.2.2 BSM Entropy Generators

In this section, we introduce a new BSM primitive which we call a BSM entropy generator (EG). We also show how to construct it using averaging samplers.

[2]Recall that in bounded storage model, the source X is typically envisioned to be a huge random object which can neither be accessed nor stored in its entirety.

Definition 7 EG*:* $\{0,1\}^n \times \{0,1\}^r \to \{0,1\}^t$ *is an* (ε, ψ)*-secure BSM entropy generator for storage rate* β *and min-entropy rate* α *if and only if for every* αn*-source* X *on* $\{0,1\}^n$ *and for every function* A*:* $\{0,1\}^n \to \{0,1\}^{\beta n}$*, the random variable* $(\mathrm{EG}(X,K), A(X), K)$ *is* ε*-close to* $(W, A(X), K)$*, where* $K \stackrel{R}{\leftarrow} \{0,1\}^r$ *and* W *is a* ψ*-source.*

Theorem 5 *Let* $1 \geq \delta - (1/n)\log(1/\varepsilon) \geq 3\tau > 0$*. Suppose* Samp*:*$\{0,1\}^r \to [n]^t$ *is a* (μ, θ, γ) *averaging sampler for* $\mu = (\delta - (1/n)\log(1/\varepsilon) - 2\tau)/\log(1/\tau)$ *and* $\theta = \tau/\log(1/\tau)$*. Define* EG*:* $\{0,1\}^n \times \{0,1\}^r \to \{0,1\}^t$ *such that* $\mathrm{EG}(X,R) = X_{\mathrm{Samp}(R)}$*. Then* EG *is a* $(\varepsilon + \gamma + 2^{-\Omega(\tau n)}, (\delta - (1/n)\log(1/\varepsilon) - 3\tau)t)$*-secure BSM entropy generator for storage rate* β *and min-entropy rate* $\beta + \delta$*.*

Proof. We shall build upon the following lemma given in [Vad04]. This is a refinement of the fundamental Nisan-Zuckerman lemma [NZ96].

Lemma 2 ([Vad04]) *Let* $1 \geq \delta \geq 3\tau > 0$*. Suppose that* Samp :$\{0,1\}^r \to [n]^\ell$ *is an* (μ, θ, γ) *averaging sampler with distinct samples for* $\mu = (\delta - 2\tau)/\log(1/\tau)$ *and* $\theta = \tau/\log(1/\tau)$*. Then for every* δn*-source* X *on* $\{0,1\}^n$*, the random variable* $(U_r, X_{\mathrm{Samp}(U_r)})$ *is* $(\gamma + 2^{-\Omega(\tau n)})$*-close to* (U_r, W) *where for every* $a \in \{0,1\}^r$*, the random variable* $W|_{U_r} = a$ *is a* $(\delta - 3\tau)\ell$*-source.*

Let A: $\{0,1\}^n \to \{0,1\}^{\beta n}$ be any arbitrary function and let X be a reverse block source of min-entropy $(\beta + \delta)n$. As in [NZ96], with probability at least $(1 - \varepsilon)$ over $S \stackrel{R}{\leftarrow} A(X)$, the random variable $X|_{A(X)} = S$ is a $(\beta + \delta)n - \beta n - \log(1/\varepsilon)$ source. Putting $\delta' = (\delta - (1/n)\log(1/\varepsilon))$, we get that $X|_{A(X)=S}$ is a $\delta' n$-source. Hence, we have $1 \geq \delta' \geq 3\tau > 0$, $\mu = (\delta' - 2\tau)/\log(1/\tau)$ and $\theta = \tau/\log(1/\tau)$. Applying the above lemma, $(U_r, X_{\mathrm{Samp}(U_r)})$ is $(\gamma + 2^{-\Omega(\tau n)})$-close to (U_r, W) where the random variable W is a $(\delta' - 3\tau)t$-source. Hence, with probability at least $(1 - \varepsilon)$, $(EG(X,K), A(X), K)$ is $(\gamma + 2^{-\Omega(\tau n)})$-close to $(W, A(X), K)$ where $K \stackrel{R}{\leftarrow} \{0,1\}^r$ and $W|_K$ is a $(\delta - (1/n)\log(1/\varepsilon) - 3\tau)t$-source. Thus, EG is a $(\varepsilon + \gamma + 2^{-\Omega(\tau n)}, (\delta - (1/n)\log(1/\varepsilon) - 3\tau)t)$-secure BSM entropy generator for storage rate β and min-entropy rate $\beta + \delta$. □

We note that a BSM PRG can be obtained from any BSM EG by applying a strong extractor to the latter's output. We also note that BSM PRG is a special case of BSM EG where the output distribution is close to the uniform distribution. Hence it is always possible to use a BSM PRG in place of a BSM EG. However, it is desirable to use the fast "sampler only" based BSM EGs at several places in our protocols where the efficiency is critical and uniformity of the output is not required.

In this paper, all our constructions of BSM EG will be sampler based. Thus the number of bits read from the source will be optimal, e.g., equal to the output size t.

D.3 Specific Instantiations of BSM PRG and BSM EG

In this section, we provide a concrete instantiation of each of the two BSM primitives discussed. We shall primarily be interested in making the primitives as fast as possible. This is in contrast to the usual goal of minimizing the input randomness and extracting as much entropy from the input source as possible.

In our setting, a particularly attractive choice as a sampler is to just uniformly pick a set of positions of the input source X. This, though is expensive on the randomness used, is very efficient and in fact requires no computations at all.

We now instantiate the BSM entropy generator EG: $\{0,1\}^n \times \{0,1\}^r \to \{0,1\}^t$ with the above sampler. We let $\mu = (\delta - 2\tau)/\log(1/\tau)$ and $\theta = \tau/\log(1/\tau)$. Each possible input to any function f in the definition of the averaging sampler is a random variable that takes values on $[0,1]$. Thus we get directly from the Chernoff-Hoeffding bound that

$$\Pr_{(i_1,\cdots,i_t)\stackrel{R}{\leftarrow}\mathrm{Samp}(U_r)}\left[\frac{1}{t}\sum_{j=1}^{t} f(i_j) < (\delta-2\tau)/\log(1/\tau) - \tau/\log(1/\tau)\right] \leq 2^{\frac{-t\tau}{\log(1/\tau)}}.$$

Thus $\gamma = 2^{\frac{-t\tau}{\log(1/\tau)}}$. Note that in this case, $r = t\log(n)$. Now, let the adversarial storage rate be β and min-entropy rate be $\beta + \delta$. Setting $\mathrm{EG}(X,R) = X_{\mathrm{Samp}(R)}$ for the above uniform t-set sampler and using Theorem 5, we get that EG is a $(\varepsilon,\psi) = (a + 2^{-t\tau/\log(1/\tau)} + 2^{-\Omega(\tau n)}, (\delta - (1/n)\log(1/a) - 3\tau)t)$-secure BSM entropy generator. Selecting $a = 2^{-(\delta/8)n}$ and $\tau = \delta/8$, we get $\psi = (\delta/2)t$. Now, given the security parameter κ, we set $r \geq (2/\delta)\kappa\log(n)$ (or $t \geq (2/\delta)\kappa$). Thus, given $A(X)$ with $|A(X)| \leq \beta n$ and R, the probability p of an adversary successfully guessing the output $y = \mathrm{EG}(X,R)$ can be bounded as follows

$$\begin{aligned} p &\leq \varepsilon + 2^{-\psi} \\ &\leq 2^{-(\delta/8)n} + 2^{-t\tau/\log(1/\tau)} + 2^{-\Omega(\tau n)} + 2^{-(\delta/2)t} \\ &\leq \mathrm{neg}(\kappa) \qquad (\text{since } n \geq t \geq \kappa) \end{aligned}$$

We now turn towards the instantiation of the BSM pseudorandom generator PRG: $\{0,1\}^n \times \{0,1\}^{r+d} \to \{0,1\}^m$. We construct it by simply applying a strong extractor to the output of the BSM EG constructed above. An attractive choice here is to use a universal hash function, which in the light of leftover hash lemma ([HILL99], [BBR88]) can be seen as a strong extractor. The leftover hash lemma is given.

Lemma 3 ([HILL99]) *Let $D : \{0,1\}^t$ be a probability ensemble that has min entropy at least m_e. Let ε be a positive integer valued parameter. Let $h : \{0,1\}^r \times \{0,1\}^d \to \{0,1\}^m$ be a universal hash function such that $m = m_e - 2\varepsilon$. Let $W \stackrel{D}{\leftarrow} \{0,1\}^t$. Then*

$$\Delta((h(W,U_d),U_d),(U_m,U_d)) \leq 2^{-(\varepsilon+1)}.$$

We let $\mathrm{PRG}(X,R) = h(\mathrm{EG}(X,R_1),R_2)$ where $R = R_1 \circ R_2$ and EG be a (ε,ψ)-secure BSM EG as constructed before using uniform t-set samplers. Setting $m_e = (\delta/2)t$ $(=\psi)$ and $\varepsilon = (\delta/8)t$, we get $m = (\delta/4)t$. Using leftover hash lemma, this yields a $(\varepsilon + 2^{-(\delta/8)t+1})$-secure BSM PRG. Given $A(X)$ with $|A(X)| \leq \beta n$ and R, clearly the probability P of an adversary successfully guessing the output $y = \mathrm{PRG}(X,R)$ is again negligible in the security parameter κ.

Hence, given the reverse block entropy source $X \in \{0,1\}^n$ of min-entropy rate $(\beta + \delta)$ and adversary storage bound β, the above PRG outputs $(\delta/4)t$ bits close to uniform using $(t\log(n) + d)$ random bits.

D.4 Other Extractors

The extractor used in the instantiation above is a fast extractor, i.e., a universal hash function. However, at a few places in our protocols, the extractor phase of the BSM pseudorandom generator (constructed using samplers and extractors) is performed offline without timing constraints. In such cases, it is desirable to use an extractor with good asymptotic performance in order to preserve the entropy of the source. Below, we briefly note a few suggestions for such extractors.

From [Zuc97] we have:

Lemma 4 *Let $\delta, k > 0$ be arbitrary constants. For every $n \in N$ and every $\varepsilon > exp(-n/2^{O(\log^* n)})$, there is a explicit strong $(\delta n, \varepsilon)$-extractor Ext:$\{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$ with $d = O(\log n + \log(1/\varepsilon))$ and $m = (1-k)\delta n$.*

There are extractors from [Tre01, RRV02] that also extract a constant fraction of the min-entropy and may be employed profitably as well.

E Secure positioning in 3-dimensional space

To securely position a prover in the 3-dimensional space, we shall use 4 verifiers denoted by $V_1, \cdots, V_4$. We assume that the location P being claimed by the prover is enclosed in the tetrahedron defined by these 4 verifiers. The verifiers V_1, V_2 and V_3 are assumed to possess reverse block sources of minimum entropy $(\delta + \beta)n$, where βn denotes the total adversary information retrieval bound. Let $t_1, \cdots, t_4$ be the time taken for radio waves to reach the point P from verifier $V_1 \cdots V_4$ respectively. When we say that V_1, V_2, V_3, V_4 broadcast messages such that they "meet" at P, we mean that they broadcast these messages at time $T - t_1, T - t_2, T - t_3$ and $T - t_4$ respectively so that at time T all these messages are at position P in space. We use a BSM pseudorandom generator namely an ε-secure PRG:$\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^m$. We choose the parameters such that $\varepsilon + 2^{-m}$ is negligible in the security parameter. X_i denotes a string drawn at random from a reverse block entropy source. We first assume that verifiers can store these X_i values. Later, we show how this assumption can be removed. The protocol is given below:

1. V_2, V_3 and V_4 broadcast strings X_1, X_2 and X_3, drawn at random from their respective reverse block entropy sources at times $T - t_2, T - t_3$ and $T - t_4$. V_1 broadcasts key $K_1 \stackrel{R}{\leftarrow} \{0,1\}^m$ at time $T - t_1$. V_1, V_2, V_3 and V_4 pre-compute $K_{i+1} = \mathrm{PRG}(X_i, K_i), 1 \leq i \leq 3$ from the stored X_1, X_2 and X_3.

2. At time T, the prover at position P computes messages $K_2 = \mathrm{PRG}(X_1, K_1)$, $K_3 = \mathrm{PRG}(X_2, K_2)$ and $K_4 = \mathrm{PRG}(X_3, K_3)$ in that order. The prover returns K_4 to all verifiers.

3. All verifiers check that the string K_4 is received at time $(T + t_i)$ and that it equals the K_4 that they pre-computed. If these verifications succeed, the position claim of the prover is accepted and it is assumed to be indeed at position P. Otherwise, the position claim is rejected.

The completeness follows from the fact that verifiers can compute K_4 from the stored X_i values and the prover can also compute K_4 since all the information required is present jointly at P at time T. The security of this protocol is proven using techniques from the proof of security of the protocol for 3-dimensional position based key exchange that is discussed in Section 7 (note that position based key exchange implies a protocol for secure positioning).

We now describe, how to remove the assumption that verifiers can store strings drawn at random from their respective reverse block entropy sources. The idea for this is as follows. Note that the problem we face when verifiers cannot store the large strings is that verifiers have no way of verifying the response of the prover. This is because, when for example, V_3 broadcasts string X_2, it does not know the key K_2 used to compute K_3 from X_2. We get around this problem as follows. The verifiers pre-determine the keys K_1, K_2, K_3, K_4 that are to be used at every iteration of the application of the PRG. Now, the expected response of the prover, K_4 is known before protocol execution to all verifiers. The protocol is as below:

1. V_1, V_2, V_3 and V_4 pick keys K_1, K_2, K_3 and K_4 drawn at random from $\{0,1\}^m$ and broadcast them over their private channel.

2. In order to enable the device at P to compute K_i for $1 \leq i \leq 4$, the verifiers do as follows. V_1 broadcasts key K_1 at time $T - t_1$. V_2 broadcasts X_1 at time $T - t_2$ and simultaneously also broadcasts $K_2' = \mathrm{PRG}(X_1, K_1) \oplus K_2$. Similarly, V_3 broadcasts $(X_2, K_3' = \mathrm{PRG}(X_2, K_2) \oplus K_3)$ at time $T - t_3$ and V_4 broadcasts $(X_3, K_4' = \mathrm{PRG}(X_3, K_3) \oplus K_4)$ at time $T - t_4$.

3. At time T, the prover at position P computes messages $K_{i+1} = \mathrm{PRG}(X_i, K_i) \oplus K_{i+1}'$ for $1 \leq i \leq 3$. The prover returns K_4 to all verifiers.

4. All verifiers check that the string K_4 is received at time $(T + t_i)$ and that it equals the K_4 that they pre-picked. If these verifications succeed, the position claim of the prover is accepted and it is assumed to be indeed at position P. Otherwise, the position claim is rejected.

The completeness of this protocol is as follows. Note that since the verifiers picked K_4 before the execution of the protocol, they can verify the response of a prover without storing any of the large random strings. To argue security of the protocol, note that in this protocol, instead of using the output of the PRG as an input key in the next round, one treats the output as one secret share of the key to be used. The other share of this key is broadcast in the clear. Now, if one of the shares of an additive secret sharing scheme is random, then the secret is hidden. Hence, by the security of the protocol in which verifiers could store the large random strings, it follows that this protocol is also secure.

F Computational Position Based Key Exchange

In this section, we show how to compile any 1-round *information theoretically* secure positioning protocol SP in our bounded retrieval model along with any unauthenticated key-exchange protocol KE into an authenticated computational position based key exchange protocol CKE in the bounded retrieval model.

Let the verifiers generate a public key - secret key pair (pk, sk) of a CCA2 secure public key encryption scheme [RS91, DDN00]. Let $\mathsf{E}(pk, m)$ denote an encryption of a message m with public key pk and let $\mathsf{D}(sk, c)$ denote the decryption of ciphertext c with secret key sk. Let KE be an unauthenticated key exchange protocol. Our protocol for computational position based key exchange CKE is as follows:

1. First, the device and an arbitrary verifier V_j carry out the unauthenticated key exchange protocol KE to obtain shared key k. Let the entire transcript of this protocol be denoted by T_k.

2. At the end of this protocol, verifier V_j sends T_k to all other verifiers through the private channel.

3. The verifiers then carry out the first message of the secure positioning protocol SP with the device at P. Let the response which the device needs to broadcast to verifier V_i be K_i for all i.

4. The device at point P now computes $c_i = \mathsf{E}(pk, T_k || K_i)$. The device broadcasts c_i to verifier V_i for all i.

5. Verifier V_i computes $\mathsf{D}(sk, c_i)$ and parses the obtained plaintext as $T_k' || K_i'$. If the response K_i' for all verifiers completes the secure positioning protocol SP successfully, then all the verifiers check if $T_k' = T_k$, and if so accept k as the shared secret key. Otherwise, the verifiers reject the response.

The high level intuition behind the security of this protocol is as follows. First, note that the above protocol CKE is also an information theoretically secure positioning protocol. This is because if an adversary can respond with $\mathsf{E}(pk, T_k||K_i)$ to V_i in CKE, then there exists a (computationally unbounded) simulator that can respond with K_i to V_i in SP. Now, by the security of SP, it follows that for an adversary to be successful in the CKE protocol, there must also be an honest prover at P sending out an encrypted message in the last round. Note that by the security of KE, it follows that an adversary which is passive during the execution of KE between the prover and the verifiers does not obtain any information about key k. Now, an *active* adversary in KE must modify the transcript T_k (as observed by the prover) to T'_k. This means that the adversary must also maul the encryption c_i to succeed in the protocol and this breaks the CCA2 security of E.

Theorem 6 *Let* E *be a CCA2 secure public key encryption,* KE *be a secure unauthenticated key exchange protocol and* SP *be a* 1*-round information theoretically secure positioning protocol against active adversaries. Then,* CKE *described above is a secure position based key exchange protocol in 3-dimensional space.*

Proof. We first begin by observing that CKE is an information theoretically secure positioning protocol. This is because, if there exists an adversary that can respond with $c_i = \mathsf{E}(pk, T_k||K_i)$ to verifier V_i within the appropriate time such that V_i accepts c_i as a correct response in the CKE security game, then there exists a (computationally unbounded) simulator that can obtain K_i from c_i and respond with K_i to V_i within the appropriate time in the SP security game and succeed with the same probability.

Now, from the fact that CKE is an information theoretically secure positioning protocol, it follows that no set of adversaries can succeed in the CKE protocol used as a secure positioning if there does not exists a prover at P interacting with the verifiers and if the adversaries do not make use of information output by the prover at P. Now, if the set of adversaries are passive, and do not modify any messages in the KE protocol, then by the security of unauthenticated key exchange protocol KE, it follow that no passive computationally bounded set of adversaries have any advantage in predicting any information about shared secret key k.

We now turn to active adversaries who modify messages in the KE protocol. We now show that if there exists a PPT adversary that can successfully modify a message in the CKE protocol and yet make the verifiers accept, then there exists a (computationally unbounded) simulator that can succeed in the secure positioning protocol SP without being present at position P.

In the protocol SP, let m_i be the message sent by verifier V_i to the device at P and let K_i be the response that the device at P must return to V_i within the appropriate time. The set of verifiers in the security game of CKE is also $\{V_1, \cdots, V_r\}$. Let $A_1, \cdots, A_l$ be the set of adversaries in the security game of CKE. Without loss of generality, we can assume that A_i responds with $\mathsf{E}(pk, T_k||K_i)$ to verifier V_i. For our proof, we assume that there exists simulator S_i at the same geographical position as A_i for all $1 \leq i \leq l$ and there exists simulator S_{V_i} at the same geographical position as V_i for all $1 \leq i \leq r$. We now consider the following hybrid experiments.

Hybrid 1: In the first hybrid, in addition to the above simulator, there exists a simulator S at position P. Initially, one of the simulators at the position of a verifier, say S_{V_j}, playing the role of the verifier V_j, executes the key exchange protocol KE with the simulator at S. Any adversary on the line between S_{V_j} and P can modify messages in this protocol and the resulting transcript of this protocol is T'_k.

The simulators pick a (pk, sk) pair of the CCA2 encryption scheme. Now, the simulators, using the verifiers $\{V_1, \cdots, V_r\}$ of the SP security game, execute the SP part of the CKE protocol. S_i, upon receiving any message m from a verifier, forwards the message to A_i. A_i may modify bits of this message to obtain modified message m^*. Note that this happens instantaneously. S_i then sends out message m^* as the modified message. Simulator S, at position P, upon receiving all the (possibly modified) messages from the verifiers, computes $c_i = \mathsf{E}(pk, T'_k||K_i)$ and broadcasts the responses. Again S_i, upon receiving c_i, forwards it to A_i. A_i, responds with (a possibly modified) c_i^*. Note that this also happens instantaneously. S_i decrypts c_i^* to obtain T'_k and K'_i. S_i sends K'_i to verifier V_i.

Clearly, this is the real game of CKE and the probability with which the simulators succeed in the SP security game is the same as the probability with which the set of adversaries succeed in the CKE security game.

Hybrid 2: In the second hybrid, we do exactly as above, except that simulator S, instead of responding with $c_i = \mathsf{E}(pk, T'_k||K_i)$ to V_i, responds with $c_i = \mathsf{E}(pk, \mathfrak{d})$ where $\mathfrak{d}$ denotes a random dummy message. Again S_i, upon receiving c_i, forwards it to A_i. A_i, responds with (a possibly modified) c_i^*. Note that this also happens instantaneously. S_i decrypts c_i^* to obtain T'_k and K'_i. S_i sends K'_i to verifier V_i.

It follows from the CCA2 security of the encryption E that, assuming a computationally bounded adversary, the probability of success of the adversary in CKE remains negligibly close to the one in the previous experiment.

Hybrid 3: In the third hybrid, we do exactly as above, except that no simulator S exists at position P. S_i executes KE with an imaginary simulator S by computing replies of S on its own and delaying messages to A_i appropriately to finally obtain the possibly modified transcript T'_k. Now, instead of S sending an encryption $c_i = \mathsf{E}(pk, K_i)$ to A_i (as the response of S at P), S_i sends $c_i = \mathsf{E}(pk, \mathfrak{d})$ to A_i (Note that the simulators can before hand agree upon the randomness to be used in the protocol so that each S_i can be consistent in its messages to A_i.). A_i responds with c_i^*. S_i decrypts c_i^* to obtain T'_k and K'_i. S_i sends K'_i to verifier V_i.

Clearly the probability of the adversary succeeding in this hybrid is negligibly close to the one in the previous hybrid since the responses and timings of the simulators are consistent with those that would have been sent by a simulator S at P.

In the third hybrid, note that no adversary exists at position P and yet the verifiers accept in the positioning protocol SP with non-negligible probability (if the adversaries succeed in the security game of CKE with non-negligible probability). The theorem follows from the security of the positioning protocol. □

Protocol without assuming public key infrastructure. We now describe our protocol for computational position based key exchange that does not assume the existence of public key infrastructure. We show how to compile any 1-round *information theoretically* secure positioning protocol SP in our bounded retrieval model along with any unauthenticated key-exchange protocol KE into an authenticated computational position based key exchange protocol CKE in the BRM.

Let $c = \mathsf{C}(m; r)$ denote a non-interactive perfectly binding commitment of message m using randomness r. Let $\mathsf{O}(c)$ denote the opening of a commitment c (namely (m, r)). Let KE be an unauthenticated key

exchange protocol. Let ZK denote a non-malleable zero knowledge proof of knowledge system. Our protocol for computational position based key exchange CKE is as follows:

1. First, the device and an arbitrary verifier V_j carry out the unauthenticated key exchange protocol KE to obtain shared key k. Let the entire transcript of this protocol be denoted by T_k. At the end of this protocol, verifier V_j sends T_k to all other verifiers through the private channel.

2. The verifiers then carry out the first message of the secure positioning protocol SP with the device at P. Let the response which the device needs to broadcast to verifier V_i be K_i for all i.

3. For all i, the device at point P computes $c_i = \mathsf{C}(T_k||K_i; r_i)$ and broadcasts it to V_i.

4. For all i, the device next gives a non-malleable zero-knowledge proof of knowledge (to the respective verifiers) using ZK for the following statement: "There exists $T_k||K_i$ and r_i such that $c_i = \mathsf{C}(T_k||K_i; r_i)$". The zero-knowledge protocols can be run sequentially.

5. The last message of the last ZK protocol must be received by the verifier at some time T' and this T' is publicly known to all parties, including the prover. After time T', the device sends $\mathsf{O}(c_i) = (T'_k||K'_i, r_i)$ to V_i.

6. The verifiers check if the c_i values were received in time by V_i, if the ZK protocol finished within time T' and if the opening of c_i is to message $T'_k||K'_i$ such that K'_i completes the secure positioning protocol SP successfully. If this is the case, then all the verifiers check if $T'_k = T_k$, and if so accept k as the shared secret key. Otherwise, the verifiers reject the response.

The correctness of the protocol is straightforward. The high level intuition behind the security of this protocol is as follows. Note that the above protocol is also an information theoretically secure positioning protocol. This is because if a prover can respond with $c_i = \mathsf{C}(T_k||K_i; r_i)$ to V_i in CKE within the appropriate time, then there exists a (computationally unbounded) simulator that can respond with K_i to V_i in SP (by breaking the perfectly binding commitment scheme). By the security of SP, it follows that a successful adversary in the CKE protocol, must make use of an honest prover at P. Note that by the security of KE, it follows that a passive adversary does not obtain any information about key k. Now, an active adversary in KE must modify the transcript T_k to T'_k. This means that the adversary must change the commitment c_i (because the perfectly binding commitment is opened later on). Now, if the adversary is to succeed later on and open the new commitment to a message containing K_i, then he must maul the zero-knowledge proof to succeed in the protocol and this breaks the non-malleability of the zero-knowledge proof system. A full proof of security for this protocol will be provided in the final version.

G Information Theoretic Position Based Key Exchange

We shall directly present a protocol to perform position based key exchange in the 3-dimensional space. We shall use 4 verifiers denoted by V_1, V_2, V_3 and V_4. Let these verifiers be at coordinates $(0,0,0), (x_2,0,0), (x_3,y_3,0)$ and (x_4,y_4,z_4) respectively in 3-dimensional space. V_1, V_2, V_3 and V_4 are such that no three of them are co-linear and all of them are not co-planar. Let the position claimed by the device be P at (a,b,c). We note here that key exchange can be done only in specific regions in space. In

Lemma 5, we characterize exactly this region. Let P be at a distance d_i from V_i and let (a, b, c) satisfy the inequality given by Equation 1.

All the verifiers are assumed to possess reverse block sources of minimum entropy $(\delta+\beta)n$ Let $t_1, \cdots, t_4$ be the time taken for radio waves to reach the point P from verifier $V_1, \cdots, V_4$ respectively.

Let PRG:$\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$ be an ε-secure BSM pseudorandom generator. K_1 is a key drawn from U_m and $K_{i+1} = \mathrm{PRG}_i(X_i, K_i)$ for $1 \leq i \leq 5$. We choose the parameters such that $5\varepsilon + 2^{-m}$ is negligible in the security parameter. We shall first assume that verifiers can store X_i sampled from the entropy sources and then show how to remove the assumption.

Let M_1 be the message pair (K_1, X_4) where K_1 is a secret key drawn uniformly at random from $\{0,1\}^m$. Let M_2 be the message pair (X_1, X_5), let M_3 be the message X_2 and M_4 be the message X_3.

G.1 Key-exchange protocol description

1. V_i broadcasts M_i in such a way that messages M_1, M_2, M_3 and M_4 reach P at exactly the same instance of global time. In other words, let T be the global time at which prover at P should receive M_1, M_2, M_3 and M_4. Then V_i broadcasts message M_i at time $T - t_i$.

2. The prover at P upon receiving all four messages, computes $K_{i+1} = \mathrm{PRG}_i(X_i, K_i)$ for $1 \leq i \leq 5$.

3. K_6 is the computed shared secret key.

We now proceed to the proof of security for this protocol. Our proof relies on work from [DP07] on intrusion-resilient random secret sharing. We first present the required background on this work.

G.2 Intrusion Resilient Random Secret Sharing Schemes [DP07]

Our main proof is based on the work from [DP07] on Intrusion-Resilient Random-Secret Sharing (IR-RSS). IRRSS is a secret sharing scheme (of an unknown random message) for a set of players ($\mathcal{P} = \{P_0, \cdots, P_{a-1}\}$). There is a dealer who sends "shares" of the message to P_i for all i. The adversary can corrupt a player and look at the memory of this player. The scheme is in the bounded-retrieval model ([CLW06, Dzi06a]) and hence assumes that an adversary can compute and store any arbitrary function on the memory of the player, but has a bound on the number of bits that he can retrieve. The scheme is guaranteed to be secure (in other words the secret is indistinguishable from random to the adversary) even if the adversary were to have temporary access to each of the shares of the players. One can consider the corruptions coming in rounds, where the adversary can choose some player to corrupt and some function to compute on that player's share in each round.

Consider the set of players $\mathcal{P}$. The IRRSS will be an a-out-of-a-secret sharing scheme. There is a dealer who creates large shares (X_i) such that each player P_i receives $X_i \in \{0,1\}^n$. The random secret will be the output of a function that requires the players to make ℓ "loops" in order to compute the secret. A loop is a sequence of a short messages sent from player P_i to player $P_{i+1 \bmod a}$, starting and ending with player P_0.

In an IRRSS an adversary will be able to adaptively issue corruption requests. However, we imagine the shares of each player are so large that while an adversary may have temporary access to a players share they cannot record the entire share. This situation models a player keeping the large random share on

their machines. If an adversary temporarily corrupts a machine, it may not be able to retrieve the share completely. The properties of the IRRSS will require that for an adversary to compute the secret share, the adversary will need to corrupt the machines in a sequence that contains ℓ loops through the players machines making the exact secret recovery process. We first introduce some notation.

Definition 8 *Let $||$ denote concatenation. Let $X = (x_1, x_2, \cdots, x_n)$ be a sequence. We say that X is a* subsequence *of sequence Y, if there exists (possibly empty) sequences $Y_1, Y_2, \cdots, Y_{l+1}$ such that $Y = Y_1||x_1|| \cdots ||Y_l||x_n||Y_{l+1}$. We also say that Y contains X.*

U_m denotes the uniform distribution on $\{0,1\}^m$ and $\delta(X, Y)$ denotes the statistical distance between distributions X and Y. Below we give the functional definition of an IRRSS from [DP07].

Definition 9 *An* intrusion-resilient random secret sharing (IRRSS) *scheme $\Xi_{a,\ell}$ is a protocol between a dealer and the players in some set $\mathcal{P}$. It consists of the following two algorithms, indexed by the number of players $a \in N$ and some parameter $\ell \in N$.*

- *$\text{share}_{a,\ell}$ is a randomized algorithm that returns a sequence $X_1, \cdots, X_a$ of bit-strings, where each X_i is of length n. The algorithm is executed by the dealer that later sends each X_i to player P_i.*
- *$\text{reconstruct}_{a,\ell}$ is a deterministic algorithm that takes as input a sequence $R = (X_1, \cdots, X_a)$ (produced by the share algorithm) and returns $K_{a+1} \in \{0,1\}^{m_{a+1}}$. The $\text{reconstruct}_{a,\ell}$ algorithm consists of $a\ell$ rounds, where in the ith round (for $i = 1, \cdots, a\ell$) player $P_{i \bmod a}$ sends a message to player $P_{i+1 \bmod a}$. The output is computed by P_0.*

In our use of the IRRSS scheme $\ell = 1$ and the remainder of the definitions and lemmas from [DP07] will reflect this. We shall also abuse notation and drop parameters from notation when they are fixed. We now give the adversarial model for an IRRSS from [DP07].

Adversarial Model. Let $\Xi = (\text{share}_a, \text{reconstruct}_a)$ be an IRRSS scheme as above. Consider an adversary $\mathcal{A}$ that plays the following game against an oracle Ω. The oracle runs share_a to obtain the values $X_1, \cdots X_a$. Now, the adversary can issue an (adaptively chosen) sequence $\text{corrupt}_1, \cdots, \text{corrupt}_e$ of corruption requests. Each request corrupt_i is a pair (P_{c_i}, h_i), where $P_{c_i} \in \mathcal{P}$, and $h_i : \{0,1\}^n \to \{0,1\}^{s_i}$ is an arbitrary function (given as a Boolean circuit). Note that this boolean circuit can have the history of the adversary hardwired into it as input. On input corrupt_i the oracle replies with $H_i := h(X_{c_i})$. We will say that the adversary chose a corruption sequence $\mathcal{C} = (P_{c_1}, \cdots, P_{c_w})$. An adversary $\mathcal{A}$ is a *valid* adversary if the sequence R (that was output by the reconstruction algorithm) is not a subsequence of $\mathcal{C}$. Finally $\mathcal{A}$ outputs a guess $\mathcal{K}_{a+1}$. We say that valid adversary $\mathcal{A}$ breaks the scheme with advantage ϵ if $\Pr[\mathcal{K}_{a+1} = K_{a+1}] = \epsilon$. When we consider adversaries in the IRRSS scheme, we only consider valid adversaries. We now define a βn bounded adversary.

Definition 10 *An adversary $\mathcal{A}$ is βn-bounded, if the corruption sequence $\mathcal{C} = (P_{c_1}, \cdots, P_{c_w})$ chosen by $\mathcal{A}$ satisfies the following: $\sum_{i=1}^{w} s_i \leq \beta n$, where $s_i := |H_i|$ is the length of the output of h_i.*

We define the security of an IRRSS scheme.

Definition 11 *An IRRSS scheme Ξ_a is $(\epsilon, \beta n)$-secure if every βn bounded valid adversary $\mathcal{A}$ breaks Ξ_a with advantage at most ϵ.*

For more details on Intrusion-resilient (random) secret sharing schemes, we refer the reader to [DP07].

G.3 Proof of Key-exchange protocol

We first introduce some conventions. We distinguish between the (only) two possibilities: an adversary receiving the strings broadcast by the verifiers and an adversary receiving information from other adversaries. When we say that an adversary $\mathcal{B}$ *directly receives information* from a verifier V_j , we mean that $\mathcal{B}$ obtains a broadcast that is made by verifier V_j. When we say that an adversary $\mathcal{B}$ *receives information* about bit-string X_i from an adversary $\mathcal{B}'$, very informally, we mean that $\mathcal{B}$ receives some function of string X_i from $\mathcal{B}'$ (this function could also depend upon several other strings). This can be defined recursively in a more precise way. An adversary $\mathcal{B}$ can receive information about X_i from an adversary $\mathcal{B}'$ only if $\mathcal{B}'$ had received information about X_i (either directly from the verifiers or from another adversary) by the time it sends the information to $\mathcal{B}$. When we say that an adversary *receives information* about X_i, we mean that either $\mathcal{B}$ received information from an adversary $\mathcal{B}'$ or $\mathcal{B}$ directly received information from a verifier.

G.3.1 Characterization of points in 3-dimensional space

In Lemma 5 below, given the positions of 4 verifiers, we exactly characterize the regions in 3-dimensional space for which position-based key exchange is achieved. More specifically, given the geometric coordinates of the 4 verifiers in 3-dimensional space, we give a condition that the geometric coordinates of position P must satisfy so that the verifiers can verify the claim of a device at position P. We require this characterization for the following reason. Since verifiers must broadcast strings such that they all meet at P at some specific instance of time t, we require a guarantee that these broadcasts do not meet again at some other point in space at (possibly) a later instance of time. Hence, we need to characterize points P that precisely prevent this.

Lemma 5 *Let V_1, V_2, V_3 and V_4 be four verifiers in 3-Dimensional Euclidean space. Without loss of generality, we assume V_1 to be at coordinates $(0,0,0)$, V_2 at $(x_2,0,0)$, V_3 at $(x_3,y_3,0)$ and V_4 at (x_4,y_4,z_4). V_1, V_2, V_3 and V_4 are such that no three of them are co-linear and all of them are not co-planar. Let P be the point at position (a,b,c) which is within the tetrahedron enclosed by V_1, V_2, V_3 and V_4. Let the distance between V_i and P be d_i for all $1 \leq i \leq 4$.*

Let M_i be the message broadcast by V_i in such a way that messages M_1, M_2, M_3 and M_4 reach P at exactly the same instance of global time. In other words, let T be the global time at which P should receive M_1, M_2, M_3 and M_4. Then V_i broadcasts message M_i at time $T - \frac{d_i}{c} = T - t_i$ where c is the speed at which radio waves travel in air. (t_i is the time taken for any message to travel from V_i to position P.)

Furthermore, assume that the messages M_1, M_2, M_3 and M_4 contain strings drawn at random from reverse block entropy sources and cannot be re-broadcast by an adversary receiving them.

Let

$$\lambda_1 = \frac{(d_2 - d_1)}{x_2}$$

$$\lambda_2 = \frac{(d_2 - d_1)(x_3 - x_2)}{x_2 y_3} - \frac{(d_3 - d_2)}{y_3}$$

$$\lambda_3 = \frac{(d_2 - d_1)(x_4 - x_3)}{x_2 z_4} + \frac{(d_3 - d_2)(y_4 - y_3)}{y_3 z_4} - \frac{(d_2 - d_1)(x_3 - x_2)(y_4 - y_3)}{x_2 y_3 z_4} - \frac{(d_4 - d_3)}{z_4}$$

Let (a, b, c) satisfy the inequality

$$\frac{2d_1 + 2a\lambda_1 - 2b\lambda_2 - 2c\lambda_3}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 1} \leq 0 \tag{1}$$

Then P is the unique point at which messages M_1, M_2, M_3 and M_4 are all simultaneously available in 3-Dimensional space.

PROOF. Since P is at position (a, b, c), we have the following equations by computing the distance of P from each of the verifiers V_1, V_2, V_3 and V_4.

$$a^2 + b^2 + c^2 = d_1^2$$
$$(a - x_2)^2 + b^2 + c^2 = d_2^2$$
$$(a - x_3)^2 + (b - y_3)^2 + c^2 = d_3^2$$
$$(a - x_4)^2 + (b - y_4)^2 + (c - z_4)^2 = d_4^2$$

We note that the messages M_1, M_2, M_3, M_4 were broadcast from the respective verification verifiers in such a way that they reached P at exactly the same instance of global time. Suppose there exists a point A at coordinates (x, y, z) in 3-Dimensional space such that M_1, M_2, M_3, M_4 reach A at exactly the same instance of global time. In this case, we note that A must be at a distance $d + d_1$ from V_1, $d + d_2$ from V_2, $d + d_3$ from V_3 and $d + d_4$ from V_4 for some value of d. Furthermore, we note that if such an A exists, then d is positive. If $d = 0$, then clearly A is at point P and if $d < 0$ this shows the existence of a point in 3-Dimensional space that is simultaneously closer to V_1, V_2, V_3, V_4 than P is, which cannot be true. Hence, $d > 0$. Writing the equations of the distances of A from each of the verifiers gives us:

$$x^2 + y^2 + z^2 = (d + d_1)^2$$
$$(x - x_2)^2 + y^2 + z^2 = (d + d_2)^2$$
$$(x - x_3)^2 + (y - y_3)^2 + z^2 = (d + d_3)^2$$
$$(x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2 = (d + d_4)^2$$

Solving for d, we obtain either $d = 0$ or $d = \frac{2d_1 + 2a\lambda_1 - 2b\lambda_2 - 2c\lambda_3}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 1}$. From Equation 1, it follows that $d \leq 0$ and hence the only point at which M_1, M_2, M_3, M_4 are all simultaneously present is point P. □

G.3.2 Main proof of Key-exchange protocol

Theorem 7 *Let X_i $(1 \leq i \leq 5)$ be strings of length n drawn at random from the reverse block entropy source S_i as discussed earlier. Let PRG:$\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$ $(1 \leq i \leq 5)$ be an ε-secure BSM pseudorandom generator. K_1 is a key drawn from U_m and $K_{i+1} = PRG_i(X_i, K_i)$ for $1 \leq i \leq 5$.*

Let the prover be at position P in 3-dimensional space such that P satisfies the condition in Lemma 5. Let verifiers be V_1, V_2, V_3, V_4 in 3-dimensional space and let the time taken for signals to travel from V_i to P be t_i. V_1 broadcasts K_1, X_4 at time $T - t_1$, V_2 broadcasts X_1, X_5 at time $T - t_2$, V_3 and V_4 broadcast

X_2 and X_3 at times $T - t_3$, $T - t_4$. Let D_{K_6} denote the distribution of key K_6 over the randomness of $K_1, X_1, \cdots, X_5$. Then for any adversary $\mathcal{B}$ (who can control a set of adversaries, with total data retrieval bound βn, none of whom is at position P), $\delta(U_m, D_{K_6}) \leq 5\epsilon$.

PROOF. The proof at a high level will be as follows. For every adversary $\mathcal{B}$ that can distinguish key K_6 from a key drawn from U_m with advantage $> 5\epsilon$, we shall construct a βn-bounded valid adversary $\mathcal{A}$ that breaks the security of an $(5\epsilon, \beta n)$ IRRSS scheme of [DP07] with advantage $> 5\epsilon$ [3].

Consider an adversary $\mathcal{B}$ that distinguishes key D_{K_6} from U_m. This adversary obtained information from a (possibly empty) set of adversaries at various positions in 3-dimensional space and from (possibly) the verifiers directly. Let the set of adversaries at various positions in 3-dimensional space be denoted by $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_r$. Now, consider those adversaries who use information only received directly from the verifiers (V_1, V_2, V_3, V_4) to compute any information and do not receive any information from other adversaries (Note that adversaries who compute some function by themselves upon receiving information directly from the verifiers alone and then receive further information from the verifiers are also not in this set of adversaries. We will deal with such adversaries later.). Let these adversaries be denoted by $\mathcal{B}'_1, \mathcal{B}'_2, \cdots, \mathcal{B}'_p$. At a high level, we shall prove the main theorem as follows.

For every adversary $\mathcal{B}'_i$, we shall construct a corresponding valid adversary $\mathcal{A}'_i$ for the IRRSS scheme that can compute any information that $\mathcal{B}'_i$ can compute (Lemma 7). Next, given any two adversaries $\mathcal{B}_i$ and $\mathcal{B}_j$, along with their corresponding adversaries $\mathcal{A}_i$ and $\mathcal{A}_j$ in the IRRSS scheme, we shall construct a valid adversary $\mathcal{A}_{ij}$ for the IRRSS scheme that will represent the joint information that can be computed by adversaries $\mathcal{B}_i$ and $\mathcal{B}_j$ (Lemma 8). Finally, given an adversary $\mathcal{B}_i$ that takes as input the output from another adversary $\mathcal{B}_j$ (with corresponding adversary $\mathcal{A}_j$) and receives information directly from a subset of the verifiers, we show how to construct the corresponding adversary $\mathcal{A}_i$ (Lemma 11). Since $\mathcal{B}$ is just an adversary that computes information from a set of adversaries at various positions and (possibly) from the verifiers directly, these processes together will give us a recursive procedure to construct adversary $\mathcal{A}$.

The corresponding IRRSS scheme that we will construct is as follows. In our setting $a = 6$. We have the IRRSS scheme Ξ with players $P_0, \cdots, P_3$. The random secret that we will be sharing is $K_6 \in \{0,1\}^m$. As before, PRG:$\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$ $(1 \leq i \leq 5)$ is an ε-secure BSM pseudorandom generator. The IRRSS scheme Ξ is constructed as follows:

- share: Choose $K_1 \in \{0,1\}^m$ uniformly at random and sample $X_1, \cdots, X_5 \in \{0,1\}^n$ at random from their respective reverse block entropy sources. Player P_0 gets K_1 and X_4. P_1 gets X_1 and X_5. P_2 gets X_2 and P_3 gets X_3. The random secret K_6 is computed using the following procedure: For $i = 1, \cdots, 5$ let $K_{i+1} := \text{PRG}(X_i, K_i)$.
- reconstruct$(K_1, X_1, \cdots, X_5)$: The players execute the following procedure:
  1. Player P_0 sends secret K_1 to P_1; P_1 sends $K_2 = \text{PRG}(X_1, K_1)$ to P_2
  2. P_2 sends $K_3 = \text{PRG}(X_2, K_2)$ to P_3; P_3 sends $K_4 = \text{PRG}(X_3, K_3)$ to P_0
  3. P_0 sends $K_5 = \text{PRG}(X_4, K_4)$ to P_1; P_1 computes and outputs the key $K_6 = \text{PRG}(X_5, K_5)$

Lemma 6 *The IRRSS scheme Ξ is $(5\epsilon, \beta n)$-secure.*

[3] We note here that the adversaries are not computationally bounded.

PROOF. This scheme is just an IRRSS scheme as in [DP07] where P_0 holds K_1 and X_4, P_1 holds $X_1||X_5$, P_2 holds X_2 and P_3 holds X_3. P_1 outputs the final key K_6 where $K_i = \text{PRG}(X_{i-1}, K_{i-1})$. It is $(5\epsilon, \beta n)$-secure against any adversary $\mathcal{B}'$ for which $(P_0, P_1, P_2, P_3, P_0, P_1)$ is not a subsequence of the corruption sequence C' (from [DP07]). □

We now prove the main three lemmas (Lemmas 7, 8 and 11) that will prove the security of our protocol.

Lemma 7 *For every adversary $\mathcal{B}'_i$ that* directly receives information *from a subset of the verifiers $\{V_1, V_2, V_3, V_4\}$, simultaneously at point and time (L_0, T_0), (but does not receive any other information) and outputs any string λ with probability p_λ in the key-exchange protocol, there exists a valid adversary $\mathcal{A}'_i$ (with corruption sequence C'_i) in Ξ that outputs the same string λ with probability $\geq p_\lambda$.*

PROOF. Consider adversary $\mathcal{B}'_i$. From Lemma 5, we know that $\mathcal{B}'_i$ cannot simultaneously directly receive information from all verifiers. We note that P_t $(0 \leq t \leq 3)$ hold exactly the same information as the information sent by V_{t+1}. Since $\mathcal{B}'_i$ does not receive information from other adversaries, the only information it can possess is at the most information from 3 out of the 4 verifiers or in other words the corruption sequence of $\mathcal{A}'_i$ will not contain at least one P_t $(0 \leq t \leq 3)$. Without loss of generality, assume that $\mathcal{B}'_i$ does not get information from V_4. Since $\mathcal{B}'_i$ can only access one string at a time, the sequence in which $\mathcal{B}'$ accesses the strings can be written as a permutation of the elements of the set (P_0, P_1, P_2). Let $S_{B'_i}$ denote this sequence. Then the corruption sequence of $\mathcal{A}'_i$ is $S_{B'_i}$. Note that $\mathcal{A}'_i$ can compute any information computed by $\mathcal{B}'_i$ and hence Lemma 7 holds. □

We say that an adversary $\mathcal{A}_i$ is a corresponding adversary for adversary $\mathcal{B}_i$ if for adversary $\mathcal{B}_i$ that outputs any string λ with probability p_λ in the key-exchange protocol, $\mathcal{A}_i$ is a valid adversary (with corruption sequence C_i) for IRRSS Ξ and outputs the same string λ with probability $\geq p_\lambda$.

Lemma 8 *Let $\mathcal{B}_i$ and $\mathcal{B}_j$ be any two adversaries in the key-exchange protocol. Let $\mathcal{A}_i$ and $\mathcal{A}_j$ be their* corresponding *adversaries respectively. Let $\mathcal{B}_{ij}$ be any adversary in the key-exchange protocol that takes as input only the outputs of $\mathcal{B}_i$ and $\mathcal{B}_j$ (and does not directly receive information from any of the verifiers). Then there exists an adversary $\mathcal{A}_{ij}$ that is a* corresponding *adversary for $\mathcal{B}_{ij}$.*

PROOF. Let S_{B_i} and S_{B_j} be the corruption sequences of adversaries $\mathcal{A}_i$ and $\mathcal{A}_j$ respectively. We shall show how to combine these corruption sequences in order to obtain the corruption sequence of adversary $\mathcal{A}_{ij}$. We know that $P_0, P_1, P_2, P_3, P_0, P_1$ is not a subsequence of S_{B_i}. Let J_0^i be the first point in S_{B_i} that is a P_0. Let J_1^i be the first point after J_0^i in the sequence that is a P_1 and so on. (J_4^i is the first point in the sequence after J_3^i that is a P_0.) Let W_t^i be the corruption sequence between J_{t-1}^i and J_t^i (with W_0^i being the corruption sequence before J_0^i and W_5^i being the corruption sequence after J_4^i). Now, S_{B_i} can be written as follows $W_0^i||J_0^i||W_1^i||\cdots||W_4^i||J_4^i||W_5^i$. Note that a valid corruption sequence will not contain J_5^i. Similarly, S_{B_j} can also be written in the above format as $W_0^j||J_0^j||W_1^j||\cdots||W_4^j||J_4^j||W_5^j$. To combine two corruption sequences, write the new corruption sequence of $\mathcal{A}_{ij}$ as $S_{B_{ij}} = W_0^i||W_0^j||J_0^i||J_0^j||W_1^i||W_1^j||J_1^i||J_1^j||\cdots||W_5^i||W_5^j$.

We first note that if S_{B_i} and S_{B_j} are valid corruption sequences, then $S_{B_{ij}}$ is also a valid corruption sequence. We are only left with showing that $\mathcal{A}_{ij}$ can compute any information that can be computed by $\mathcal{B}_{ij}$. To see this, we note that $\mathcal{A}_{ij}$ internally can simulate the information provided by both $\mathcal{B}_i$ and $\mathcal{B}_j$ by just looking at those parts of the corruption sequence that correspond with S_{B_i} and S_{B_j} respectively.

This is because information provided by B_i and B_j to B_{ij} is bounded information and can be computed in parallel while reading the corruption sequence $S_{B_{ij}}$ by $\mathcal{A}_{ij}$. Now, $\mathcal{A}_{ij}$ can run the same algorithm that $\mathcal{B}_{ij}$ runs to output string λ, and output the same string λ with probability $\geq p_\lambda$. Hence, adversary $\mathcal{A}_{ij}$ can compute any information that can be computed by $\mathcal{B}_{ij}$ and adversary $\mathcal{A}_{ij}$ is a valid adversary (with a valid corruption sequence) in Ξ. Hence $\mathcal{A}_{ij}$ is a corresponding adversary for $\mathcal{B}_{ij}$. □

The only case we are left with is the case when an adversary $\mathcal{B}_i$ receives information from an adversary $\mathcal{B}_j$ as well as directly receives information from a verifier. Without loss of generality, we can assume that $\mathcal{B}_i$ receives information only from one adversary $\mathcal{B}_j$ as the case when $\mathcal{B}_i$ receives information from multiple adversaries can be combined first using Lemma 8. Before we discuss this case, we first present two lemmas.

Lemma 9 *In Lemma 8, let t be defined as the largest integer such that either J_t^j exists in S_{B_j} or J_t^i exists in S_{B_i}. Then, t is the largest integer such that J_t^{ij} exists in $S_{B_{ij}}$.*

Proof. The above lemma follows from the construction of $S_{B_{ij}}$ in Lemma 8. We note that the value of t can increase only if B_{ij} directly receives information from some verifier. □

Lemma 10 *Let $h(X_i, \cdot)$ be information that an adversary $\mathcal{B}_i$ receives or computes at time T_0. Then, $\mathcal{B}_i$ cannot* directly receive information X_i *from a verifier at any time $T_1 > T_0$.*

Proof. This follows from triangle inequality. □

We now consider the case when an adversary $\mathcal{B}_i$ receives information from an adversary $\mathcal{B}_j$ as well as directly receives information from a verifier.

Lemma 11 *Let $\mathcal{B}_i$ be an adversary in the key exchange protocol that* receives information *from one adversary $\mathcal{B}_j$ (with* corresponding *adversary $\mathcal{A}_j$ with corruption sequence S_{B_j}) and* directly receives information *from a subset of verifiers from the set $\{V_1, V_2, V_3, V_4\}$. Let t be the largest integer such that J_t^i exists in S_{B_i}. Then $t < 5$.*

Proof. Assume for contradiction that there exists an adversary $\mathcal{B}_i$ at some point in the key exchange protocol with $t = 5$. This means that there must have existed an adversary $\mathcal{B}_j$ at some point when $t = 3$ for the first time (There may exists many such adversaries; the following argument will apply for all such adversaries.). Consider this adversary. Note that at the position of this adversary, the string with X_3 is present at this instance of time (since $t = 3$ for the first time). Now since $t = 3$, this means that $\mathcal{B}_j$ has already seen information about K_1 and X_1. Hence, the strings with $M_1 = (K_1, X_4)$ and $M_2 = (X_1, X_5)$ were seen by $\mathcal{B}_j$ at an earlier instance of time or are present at this instance at the position where $\mathcal{B}_j$ is. If one of the strings (M_1 or M_2) were seen by $\mathcal{B}_j$ at an earlier instance, then by Lemma 10 it follows that t cannot become 5 for any adversary in the system (as any information from this adversary cannot be jointly present either with X_4 or with X_5). If both the strings M_1 and M_2 are currently present at this point, then by Lemma 10, we have that the adversary could not have seen M_1 or M_2 at an earlier instance of time. Hence, he receives information from M_1 and M_2 for the first time at this point. This means that X_2 is also present at this point at the same instance (since otherwise t cannot become 3). However, this cannot be the case by Lemma 5 and 10. Hence, t could not have been 3 at this point. Hence in $\mathcal{B}_i$, the value of $t < 5$. □

We complete the proof of security of our protocol by showing that if there exists an adversary $\mathcal{B}$ in the key-exchange protocol that can distinguish D_{K_6} from U_m, then there exists a corresponding adversary $\mathcal{A}$ in Ξ that can distinguish K_6 from a key drawn from U_m. Given an adversary $\mathcal{B}$, we construct adversary $\mathcal{A}$ recursively using the constructions in Lemmas 7, 8 and 11. Through the proofs of Lemma 7, 8 and 11 it follows that $\mathcal{A}$ can compute any information that $\mathcal{B}$ can. Since $\mathcal{A}$ runs exactly the same algorithm as $\mathcal{B}$ and accesses the same information as $\mathcal{B}$, it also follows that the total number of bits retrieved by $\mathcal{A}$ is the same as $\mathcal{B}$ which is less than βn. Therefore, $\mathcal{A}$ is a valid adversary (with a valid corruption sequence) in Ξ, and hence the probability with which $\mathcal{B}$ can distinguish D_{K_6} from U_m is 5ϵ which is negligible in the security parameter. □

We note that one can remove the assumption on verifiers storing the large random strings by using the same technique described in Section 5 on secure positioning. We can modify the above proof to argue security of this protocol. Note that in this protocol, instead of using the output of the PRG as an input key in the next round, one treats the output as one secret share of the key to be used. The other share of this key is broadcast in the clear. Now, if one of the shares of an additive secret sharing scheme is random, then the secret is hidden. Hence, by the above security proof, it follows that this modified protocol is also secure. More details will be provided in the full version of this paper.

H Identifying regions in 2-Dimensional and 3-Dimensional space where Key exchange is possible

We shall now prove that there exists a large region within three points in 2-Dimensional space such that any point P within this region is a unique point where M_1, M_2, M_3 are available simultaneously. Thus, a natural variant of our 3-dimensional protocol works for key exchange in 2-dimensions for a claimed point in that region. We shall consider the special case when the three points form an isosceles right angle triangle. Without loss of generality assume that the three verification verifiers in 2-Dimensional space are given by V_1 at $(0,0)$, V_2 at $(a,0)$ and V_3 at $(0,a)$. In the case of 2-Dimensions the corresponding condition (analogous to Equation 1 in the 3-Dimensional case) we need to check is given by the following expression:

$$\frac{(d_2-d_1)^2}{x_2^2} + (\frac{1}{x_2 y_3}(x_2(d_3-d_1)+x_3(d_2-d_1)))^2 - 1 < 0 \tag{2}$$

Substituting values of $x_2 = a, x_3 = 0$ and $y_3 = a$ in Equation 2, we obtain $(d_2-d_1)^2 + (d_3-d_1)^2 < a^2$. We shall now identify a region within the three verification verifiers where this condition is satisfied. Consider the triangle formed by the three points K_1 at (λ,λ), K_2 at $(a-\lambda,\lambda)$ and K_3 at $(\lambda, a-\lambda)$ with $\frac{a}{4} < \lambda < \frac{a}{2}$. We claim that any point P inside this triangle satisfies Equation 2.

Proof. We note that for any point P at (a,b) inside the triangle formed by K_1, K_2 and K_3, $a, b \geq \lambda$. Hence we have $d_1, d_2, d_3 \geq \sqrt{2}\lambda$. The point in the triangle at maximum distance from V_2 is K_3. Hence we have $d_2 \leq \sqrt{2}(a-\lambda)$. Similarly $d_1, d_3 \leq \sqrt{2}(a-\lambda)$. Substituting these bounds in Equation 2, we obtain $\frac{a}{4} < \lambda < \frac{3a}{4}$, which is satisfied by our assumption on λ [4]. □

Consider four verification verifiers V_1, V_2, V_3, V_4 in 2-Dimensional space at points $(0,0), (a,0), (0,a)$ and (a,a). Now consider the square bounded by four points K_1, K_2, K_3, K_4 at $(\lambda,\lambda), (a-\lambda,\lambda), (\lambda, a-\lambda), (a-$

[4] We require $\lambda < \frac{a}{2}$ so that the three points will form a triangle.

$\lambda, a-\lambda)$ with $\frac{a}{4} < \lambda < \frac{a}{2}$. For any point P inside the square bounded by K_1, K_2, K_3, K_4, P and V_2, V_3 along with either V_1 or V_4 as the third verifier satisfy the condition required.

Below we give a few examples of the geographical region within a triangle (in the 2-Dimensional case) and tetrahedron (in the 3-Dimensional case) where secure position-based key exchange is possible. The region in red marks the area where secure position-based key exchange is possible.

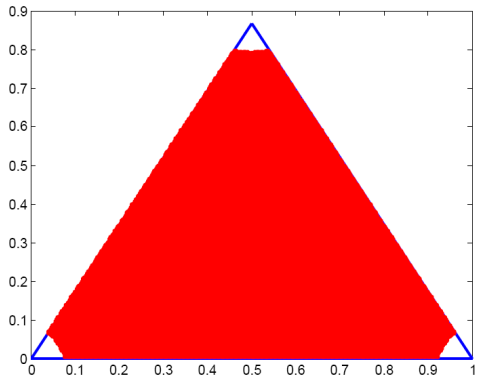

Figure 3: 2 Dimensions Equilateral Triangle

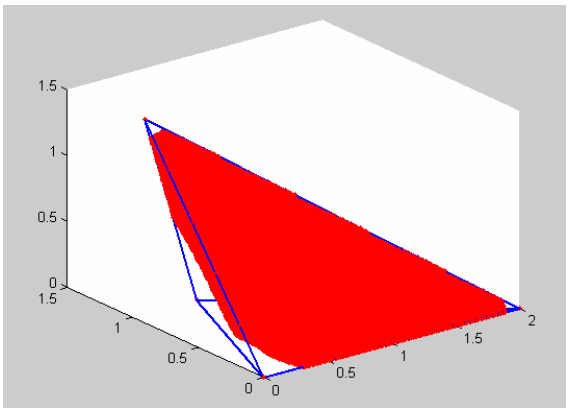

Figure 4: 3 Dimensions Example 1

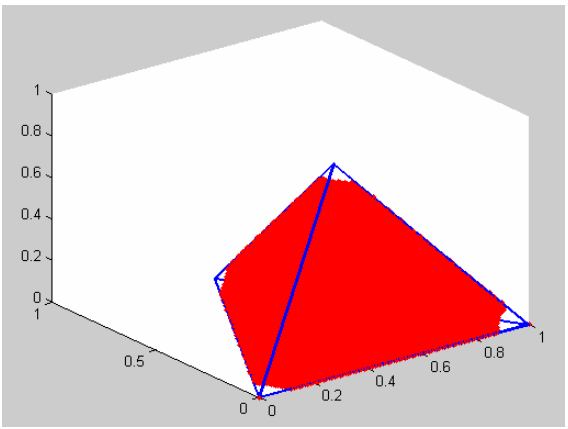

Figure 5: 3 Dimensions Example 2

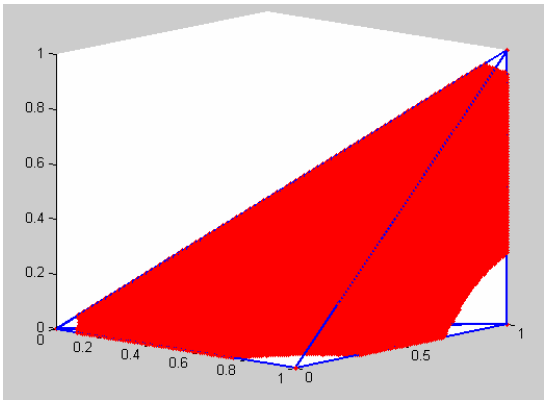

Figure 6: 3 Dimensions Example 3

I Attacks on [CCS06]

To position a device securely in the presence of multiple adversaries holding cloned devices, Capkun et al. [CCS06] propose two solutions: the first, based on "covert" (or "hidden") base stations and the second, based on mobile base stations. In the first model, there exists base stations that remain silent on the network and only observe signals. These base stations enable a verification authority to determine the position of a device. The protocol is secure under the assumption that adversaries cannot determine or "guess" the location of these hidden base stations. The second model is based on base stations that constantly move within the network. The positioning is based on the assumption that since the base stations move randomly, the adversary nodes cannot pre-determine the position of the base stations.

We now take a closer look at the above protocol [CCS06] for secure positioning in the presence of multiple adversaries. As noted above, Capkun et al. [CCS06] consider a model that makes use of base stations that may be of two types: stationary hidden base stations or mobile base stations. In stationary hidden base stations, the main assumption is that an adversary cannot determine or guess the position of the base station. In mobile base stations, the main assumption is that an adversary cannot pre-determine the position of the base station as the base station moves randomly within the network. In both models, they assume that the adversary may have directional antennas, which is precisely the model we consider (see Page 2, Section II.B: Attacker model [CCS06]). The protocols of [CCS06] are given in both infrastructure-centric service model (the infrastructure computes the position of a device) and the node-centric service model (the infrastructure verifies a position claimed by the device). In this paper, we show both the protocols in [CCS06] to be insecure in common situations. We emphasize that we are able to compromise the security of the protocols in either of these models so long as the following conditions are satisfied.

Stationary hidden base stations: In the case of stationary hidden base stations we show that the adversaries can determine the position of the hidden base stations in certain situations. In order to carry out our attack, we require that the following two conditions hold.

- First, we require that the adversaries receive feedback on whether their position was accepted or rejected by the positioning infrastructure (in the case of node-centric positioning) and that the adversaries receive their position (in the case of infrastructure-centric positioning). Indeed, if this is not the case, then it means that there is no difference between an adversary who lies and claims the wrong position and a participant that is at the right position. Thus, any protocol where there is no difference in the outcome appears to be useless. We argue that this condition is essential in many situations that we discuss below.

- The second condition that we require is that the protocol can be used several times. More specifically, we require that the adversary can make a $O(log(\frac{1}{\delta}))$ number of queries in total even if some of the queries are rejected by the infrastructure. The value δ represents the precision with which the adversary wishes to locate the hidden base station. More precisely, it is the radius within which the adversary wishes to identify that a hidden base station exists. We emphasize that we only require an order of a logarithmic number of queries (since we are able to use binary search techniques rather than exhaustive search). Indeed, protocols that can be used only a few times and then must stop operations are not particularly useful, since in general we wish to design protocols that remain

secure even if used multiple times. Below, we discuss further why this condition is essential in many applications.

If these two conditions hold, we first show that the assumption made by [CCS06] (that adversaries cannot determine the position of hidden base stations) is incorrect. We give an attack that allows the adversary to compute the position of a hidden base station with arbitrarily small error. In fact, our attack makes use of the secure positioning protocol in a black-box manner (i.e., is independent of the actual functioning of the hidden base station protocol). Therefore, it generalizes to any protocol for positioning that may be based on the assumption that base stations can be "hidden". Hence, we show that the "hidden" base station assumption is invalid and positioning protocols should not be based on this assumption.

Neither of the necessary conditions for our attacks were explicitly forbidden by the original protocols of [CCS06]. The validity of these conditions seems to be outside the scope of actual protocol and will be determined by the situation that the protocol is used in. Indeed in many common deployments of the positioning protocol, it will be the case that these conditions are inherently true.

The first condition of requiring an accept or reject be known by the adversary is reasonable in many application of the positioning protocol. One simple example is a device trying to access a resource (eg. a printer) and is given access to the resource only if it is within a specified range of the resource. An adversary will explicitly know whether his position was accepted or rejected simply by knowing whether it was granted access to the device. Another example where this condition seems inherent is using the positioning infrastructure to create an ad-hoc network. In this case the device will know whether its position was accepted by whether it becomes part of the ad-hoc network or not. More generally it seems that if this protocol is used as part of a larger protocol where the nodes will have some interaction after they are positioned, there is a strong possibility that the node will know whether its position was accepted or rejected even without getting an explicit accept or reject message.

The second condition of the adversaries needing to make a small number of queries, even if some of the queries result in "reject", is again reasonable and possibly inherent in many situations in which the protocol may be deployed. One possible "fix" to avoid this attack is to forbid a node from making any more positioning attempts if its position was rejected once (or a threshold number of times). We emphasize that this is not the protocol originally given in [CCS06] and adding this condition is already a "fix" to their protocol. However, this fix seems unreasonable in the event that a node's attempt is rejected due to communication or other errors in the system. Also if the protocol were changed as suggested, the adversary would now have an easy denial of service attack by blocking or modifying part of an honest party's signal so that the honest party fails in a positioning protocol. This attack would lead to a ban on an honest user in the system.

Furthermore, even with this fix to the original protocol given in [CCS06], it still may not be possible to limit the number of adversarial queries below the required number needed to carry out such an attack. Since the attack only requires a small number of queries, it may be the case that the attack could be carried out by a number of adversarial nodes, equal to the required number of queries (which is not unreasonable since the number of required queries is low). In such a situation each adversary would only need to make a single query and then combine their information. This is very reasonable in a situation where there are many nodes in the system or there is a high turn around on the number of nodes entering and leaving the system. We also note that these are stationary positioning stations and it may be the case that we want to use this infrastructure for a long time (e.g., years). In this situation, it seems very reasonable that an

adversary would be able to carry out the this attack.

A concrete example of where this attack would be possible is granting resource access to a device. It could easily be the case that the number of users in the system far exceeds the number of queries needed to locate the hidden base stations. In such a case, the fraction of corrupted users would only need to be small. Or if the users that are able to access the device change over time, it could be possible to locate the hidden base stations over a period of time.

These are the only conditions that are required for the attack. While it seems more natural that feedback on whether a query was accepted or rejected would be available in the node-centric model, even in the infrastructure-centric model it seems plausible that a device gets back a response on its computed position. Now, if an adversarial node is trying to spoof a particular position, then by the response received it can check if its spoofing was successful or not.

If these conditions are not met, then the first attack on positioning in stationary hidden base station model will not be successful. However we still feel it is important that these attacks be brought to the attention of the security community. These attacks throw light on some situations where the positioning protocol will be insecure. These situations seem very natural and are within the model proposed by [CCS06]. These attacks also bring to light the "fixes" that could be necessary in many situations where the infrastructure must ban a node if it gets a reject from the infrastructure. Unless someone implementing this protocol knows of this necessary fix, they may not build it into their protocol resulting in a security loophole in the system. Further, since the attack is independent of the actual positioning protocol, this shows an inherent drawback in the assumption that base stations can be "hidden" from adversaries. We show that this is not a valid assumption to make to obtain a secure positioning protocol.

Since this attack in within the model given by [CCS06], it also raises a concern that the security analysis presented in [CCS06] is not sufficient. While the conditions described above are sufficient to lead to an attack on the protocol, they may not be necessary and other security vulnerabilities may exist.

Mobile hidden base stations: In our second attack (independent of the first), we show that a set of adversaries can spoof a location in the given positioning protocol in the mobile base station model [CCS06]. Here, we show that for an adversary to have a "low" probability of spoofing a location, the number of mobile base stations required in the infrastructure is very large. This shows that the secure positioning in this model may well be impractical.

This location spoofing attack has no additional conditions for its success. For analysis, the only necessary condition is that there exists a circle (centered around the location which the adversaries want to spoof), within which the mobile base stations do not enter during the protocol execution. We believe this to be a condition that is inherently true in all interesting secure positioning protocols for the following reason. If there is no small circle that the mobile base stations do not enter, then the task of secure positioning becomes trivial; a mobile base station goes exactly to the claimed position and verifies that the device is there either by physical contact or simply verifying that it is within a small enough radius ϵ using the inherent restriction on the speed at which information can travel. We note that typically mobile base stations are envisioned as satellites and there is always a good probability that none of these satellites come too close to the location being spoofed.

Although we present this attack in the mobile base station model, we note that it can be used in the stationary base station model as well. However we presented the first attack separately because that

actually locates the hidden base stations and opens up the stationary infrastructure to many more attacks (than just this form of location spoofing).

I.1 Positioning based on Hidden Base stations

In this section, we describe the protocol for node-centric and infrastructure-centric positioning with hidden base stations proposed in [CCS06]. The system and adversary model are as outlined below:

I.1.1 Model

System model: The system consists of a set of covert base stations (CBS) and a set of public base stations (PBS) forming the positioning infrastructure. A covert base station is a base station whose position is known only to the authority controlling the verification infrastructure. Covert base stations only listen to ongoing communication; they never send any message so their position cannot be detected through radio signal analysis. The base stations know their correct positions and the adversary cannot tamper with the positions or compromise a base station. The covert base stations can measure received signal strength and can also perform ranging. Covert base stations can communicate secretly with the verification authority.

Adversary model: There are two types of positioning systems: *node-centric* and *infrastructure-centric*. In a node-centric model, the node computes its position and reports this to the authority. The authority then runs a verification procedure to check that the position reported is indeed correct. In an infrastructure-centric model, the verifying authority needs to compute the position of the node without any prior information.

We note that there are two kinds of attacks possible: *internal* and *external*. In an internal attack, a set of dishonest nodes (or adversaries), try and convince a verifying authority that some one of them exists at a position where no adversary actually is present. In an external attack, the adversary nodes try and prevent the verification authority from being able to determine or verify the position of a node in the network.

The most common approach to positioning has been that based on time difference of arrival. In this approach, the verification authority sends a nonce challenge to the device which reports this nonce back to the authority. Based on the time of response, the verification authority can verify or compute the position of the device. In this approach for positioning, an adversary can cheat on the position by sending signals to base stations at different times and by varying the strength of the radio signal. Attackers may also make use of directional antennas to perform such tasks.

I.1.2 Protocol for positioning in Hidden Base stations model [CCS06]

Capkun, Cagalj and Srivastava [CCS06] propose a method to perform infrastructure-centric and node-centric positioning of a node. The infrastructure-centric protocol is based on time-difference of arrival (TDOA) and covert base stations. TDOA is the process of positioning a source of signal in two (three) dimensions, by finding the intersection of multiple hyperbolas (or hyperboloids) based on time-difference of arrival between the signal reception at multiple base stations. The node-centric protocol is based on a single hidden base station and on the time difference in receiving radio signals and ultrasound signals.

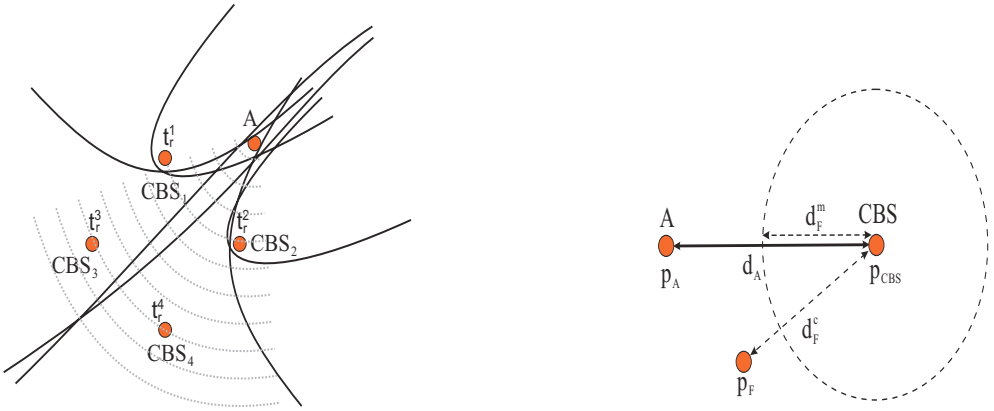

Figure 7: Infrastructure-centric and Node-centric Positioning based on covert base stations[CCS06]

The two protocols are fairly similar. The main assumption in the protocols is that the probability of an adversary "guessing" the position of a covert base station is very low.

Infrastructure-centric positioning: The public base station (PBS), at time 0, sends out a challenge nonce (N) to the node A, whose position is to be determined. The node receives this nonce at time t_s, where t_s is the time needed for radio signals to travel from the PBS to A. This node sends out a signal with this challenge value. The hidden base stations (located at various positions in the plane) note the times at which each station received the nonce. The base station, then computes the node's location with TDOA and check if this is consistent with the time differences notes (refer Figure I.1.2).

Let the covert base stations be $\{CBS_1, CBS_2, ..., CBS_k\}$ and let the time taken for radio signals to travel from A to the position of CBS_i be t_i. p is the position of node A computed from measured time differences and it is the solution to the least square problem $p = \text{argmin}_{p^*} \sum_{i>j}(|t_i - t_j| - h(p^*, i, j))^2$. $h(p^*, i, j)$ is the difference of signal reception times at CBS_i and CBS_j, if the signal is sent from position p^*. Δ is the maximum expected inconsistency between the computed position and the measured time differences. The proposed protocol is given below. For more details, we refer the reader to [CCS06].

Capkun et al. claim in [CCS06] that since the adversaries can guess the positions of base stations only with a very low probability, the TDOA with hidden base stations can detect internal and external attacks. We show that TDOA protocol is insecure under an attack made by several colluding adversaries (described in section I.2). In particular, we show that a set of adversaries can run a strategy in a way that the probability of finding the positions of the base stations is not low.

Node-centric positioning: In this system, the node reports a claimed position to the verification infrastructure. The verification authority then runs a procedure to verify if the claimed position is correct. The protocol is based on a single hidden base station and the difference between the time at which radio signals are received and ultrasound signals are received. This protocol also assumes that the adversary nodes cannot "guess" the location of a hidden base station. The public base station sends

a challenge nonce N to node A. A sends a radio signal and an ultrasound signal, both containing the claimed position p_F. The covert base station computes the distance $d_F^c = d(p_F, p_{CBS})$ and compares it the calculated position between CBS and p_F as $(t_{us} - t_{rf})s$, where t_{us} and t_{rf} are the times at which the ultrasound signal and radio signal are received respectively. If this difference between the two calculated positions is less than Δ, then the position is accepted. Refer [CCS06] for more details.

I.1.3 Assumptions in the Hidden base model

The main assumption in the hidden base model, is that the position of the covert base stations cannot be "guessed" correctly by the nodes with significant probability. If the position of the base stations are known, then the adversary can carry out collusion attacks based on timing of the signals in such a way that he can convince the verification authority that he is at a false position. In the following section, we show that the assumption that adversaries cannot guess the position of a covert base station is incorrect. In particular, we show a protocol in which a set of adversaries can determine the location of covert base station with significant probability.

I.2 Insecurity of the hidden base stations model

In this section, we show how a set of adversaries colluding together, can determine the location of a hidden base station with significant probability. Once this has been done, the attackers can carry out known attacks based on timing of the signals to break the positioning protocol. We call determining the position of a device or station upto within a radius of δ as determining the position with δ-precision. We show how a set of adversaries working in a controlled manner can locate the position of a "hidden" base station in the two-dimension plane with δ-precision in running time $O(log(\frac{1}{\delta}))$. As in [CCS06], we assume that the plane is a two-dimensional disk of radius r. For simplicity, we will first show how a set of adversaries can locate the position of a single "hidden" base station and then show how to extend this attack to multiple "hidden" base stations. As stated before, we assume that adversaries are legitimate nodes in the system, can therefore interact with the verification authority multiple times and receive `"Accept"` or `"Reject"` responses.

At a high level, the adversaries will "test" the response of the verification authority on various runs of the protocol. Based on whether the verification authority accepts or rejects a run, the adversaries will gain some partial knowledge on the location of a hidden base station. More precisely, the adversary nodes will follow the TDOA protocol from section I.1.2, with the only difference that they will now broadcast the nonce response only in certain directions (i.e, in sectors) using a directional antennae. If the verification authority accepts the positioning attempt, then certainly no CBS exists in the directions that the adversary did not broadcast the nonce. If the verification authority rejects the attempt, then there exists at least one CBS in the direction that the adversary did not broadcast the nonce (see Figure 8). Using this technique, the adversaries can conduct a binary search on the area of the two-dimensional disk to determine the location of the covert base stations. Through a geometric analysis, we then show that the distance from any node within the disk to a hidden base station can be determined with δ-precision in time $O(log(\frac{1}{\delta}))$.

Necessary conditions for the attack: In order to carry out this attack we require that the following conditions hold: 1) the adversary can determine whether his claimed position was accepted or rejected ,

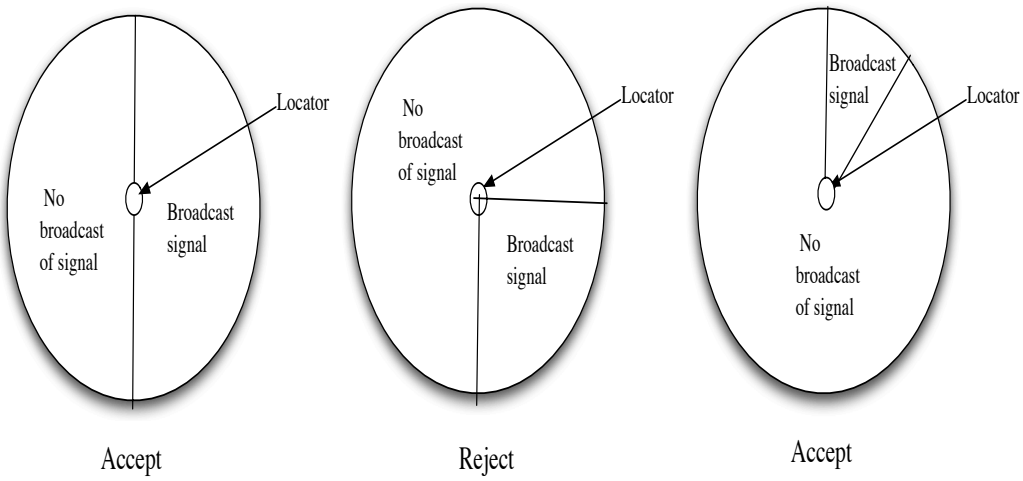

Figure 8: Binary search on the disk space to locate CBS

and, 2) the adversary can make the required number of queries. If these conditions hold than our attack will be able to located the "hidden" base stations. Again these assumptions may not only be reasonable in many situations be also inherently true in others. We refer the reader to the introduction for examples and details on the plausibility of the assumptions.

I.2.1 Determining the position of a single hidden base station

We present the formal procedure for determining the location of a single hidden base station. We assume that there are three adversary nodes (called *locators*) at positions l_1, l_2, l_3 such that l_1, l_2, l_3 do not lie on the same straight line. The nodes can communicate securely amongst themselves. The adversary nodes wish to determine the position of a base station with δ-precision. The nodes run a binary search as described in Protocol 3. We show below how the adversaries decide when a sector is "narrow enough". Let l_1 and l_2 be adversaries that return different sectors on **LOCATE-SINGLE** and let the distance between l_1 and l_2 be x. Let the angle of the sector to which each adversary broadcasts the response be γ. The two sectors, upon intersection form the quadrilateral $Q_2P_2Q_1P_1$ as shown in Figure 9. Let the length of the two diagonals of this quadrilateral be y and z. The angle between Q_1, l_1 and l_2 is α and the angle between Q_1, l_2 and l_1 is β. The two adversaries know that a hidden base station exists inside the quadrilateral $Q_2P_2Q_1P_1$. If the length of the longest chord in this quadrilateral is less than the desired error δ, then the adversaries know the position of the covert base station with δ-precision. We know that either line P_1P_2 or Q_1Q_2 must be the longest chord in this quadrilateral. We have,

$$y^2 = \left(\frac{x\sin(\gamma+\alpha)}{\sin(\alpha+\beta+2\gamma)}\right)^2 + \left(\frac{x\sin\alpha}{\sin(\alpha+\beta)}\right)^2 - 2\left(\frac{x\sin(\gamma+\alpha)}{\sin(\alpha+\beta+2\gamma)}\right)\left(\frac{x\sin\alpha}{\sin(\alpha+\beta)}\right)\cos\gamma$$

$$z^2 = \left(\frac{x\sin(\alpha)}{\sin(\alpha+\beta+\gamma)}\right)^2 + \left(\frac{x\sin(\alpha+\gamma)}{\sin(\alpha+\beta+\gamma)}\right)^2 -$$

1. Adversary l_i runs the following procedure:

   (a) Let the sector in which the covert base station may exist be π^i. Divide this sector into two sectors of equal area, π_1^i and π_2^i.

   (b) The adversary broadcasts the nonce response signal to π_1^i, but not to π_2^i using a bidirectional antenna (π_1^i is chosen arbitrarily).

   (c) If the response from the verifying authority is `"Accept"`, then set $\pi^i = \pi_1^i$, otherwise set $\pi^i = \pi_2^i$. If π^i is "narrow enough" to resemble a straight line, then go to step 2, else go to step (a).

2. Now, knowing that the covert base station lies on narrow sectors π_i and π_j for some $1 \leq i, j, \leq 3$ the adversaries can compute the position of the covert base station with δ-precision.

3. We note that this procedure will not be successful in the case that CBS, l_i and l_j lie on the same straight line (as π_i and π_j will be the same narrow sector). Hence, we need three adversaries l_1, l_2, l_3 with the condition that they are not co-linear.

Protocol 1 [LOCATE-SINGLE]: Strategy of an adversary to locate the position of hidden base stations

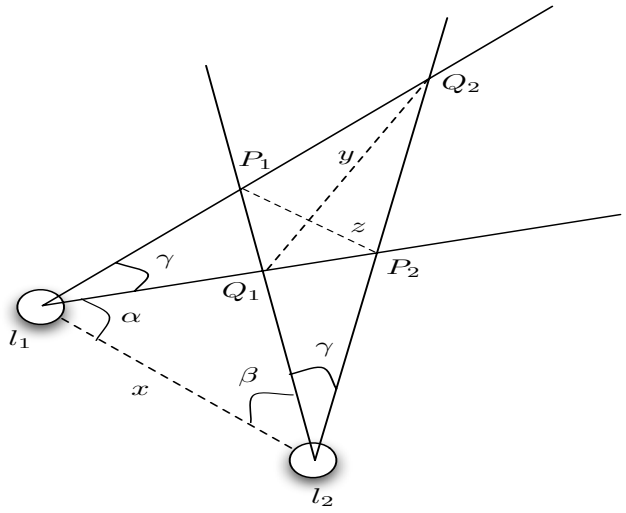

Figure 9: Finding maximum error of computing the position of CBS

$$2\left(\frac{x\sin(\alpha)}{\sin(\alpha+\beta+\gamma)}\right)\left(\frac{x\sin(\alpha+\gamma)}{\sin(\alpha+\beta+\gamma)}\right)\cos\gamma$$

The values of y and z are properly defined (as we pick adversaries l_1 and l_2 accordingly) and the larger of these two values gives us the maximum error. We now show that as γ decreases, y and z both decrease polynomially. If this is the case, then by reducing γ, we can reduce the error in computing the position of CBS to less than δ. To show that the relation between γ, y and z is polynomial, we can expand out the sine and cosine terms in the expressions for y and z using the Taylor series. Dropping higher order negligible terms gives us a polynomial relationship between y, γ and z. Thus we obtain a polynomial relationship between δ and γ. The algorithm that the adversary runs reduces γ successively by a factor of 2 and hence, we can achieve any given precision δ in $\mathrm{O}(\log(\frac{1}{\delta}))$ steps.

I.2.2 Determining the position of multiple hidden stations

When there are multiple hidden base stations, we need to modify the location procedure of the adversaries slightly. At every broadcast stage, if the adversary receives a `"Reject"` output from the protocol, it can not assume that the area broadcasted to contains no hidden base station (it only shows that there was a tower in an area not broadcasted to). Thus before the adversary determines which half the base station may be in, it will broadcast to π_1^i and then to π_2^i. If the output is `"Reject"` when it broadcasts to both halves it must be the case that there is at least one tower in each half. It will then choose one half and always broadcast to that half, while it will continue to do a binary search on the other half. Once it has located the hidden tower to enough precision in the second half it will then return to locating the tower in first half and do a binary search on that region. The adversary follows this strategy *recursively* at every stage whenever it gets a response of `"Reject"` from the verification authority. Let the total number of hidden base stations be a constant C. It is easy to see that the above procedure will then take $\mathrm{O}(\log(\frac{1}{\delta}))$ steps when δ-precision is desired.

We note that our protocols to locate the positions of multiple hidden base stations were actually independent of the positioning protocol. As long as adversaries receive a response about acceptance or rejection of the protocol that they participated in, they can locate the positions of the "hidden" base stations. This implies that the assumption that one can keep base stations "hidden" from adversaries is invalid and the security of positioning protocols should not be based on this assumption.

I.3 Positioning based on Mobile Base stations

The protocol used for positioning a device with mobile base stations [CCS06] is similar to the protocol for node-centric positioning with hidden base stations. Instead of having a fixed station whose location is unknown, we have a base station whose position changes continuously. This mobile base station sends a verification request (or nonce challenge) at one position and then waits for the response at a different position. In this way, an adversarial node cannot, ahead of time, determine the position at which the base station will receive the response. We now present the protocol more formally.

K is the secret key shared between the mobile station MBS and the node S; T_R is the time after which the node must send its reply (ideally $T_R = t_2 - t_1$). T_R is also an estimated time within which the MBS will move from one position to another. At time t_1, the mobile base station (MBS), is at position $p_{MBS}(t_1)$ and sends the challenge nonce N and a time delay T_R after which the node must reply to the

1. Adversary l_i runs the following procedure (**locate-sectors**):
   (a) Let the sector in which the covert base station may exist be π^i. Divide this sector into two sectors of equal area, π_1^i and π_2^i.
   (b) The adversary broadcasts the nonce response signal to π_1^i, but not to π_2^i using a directional antenna (π_1^i is chosen arbitrarily).
   (c) If the response from the verifying authority is `"Accept"`, then first check if π_1^i is "narrow" enough to resemble a straight line. If this is the case, then go to step 2, otherwise run **locate-sectors** on π_1^i. If the response from the verification authority is `"Reject"`, then run procedure **locate-sectors** on both π_1^i and π_2^i.
2. Now, knowing that the covert base station lies on narrow sectors π_i and π_j for some $1 \leq i, j, \leq 3$ the adversaries can compute the position of the covert base station with δ-precision. This is done for all covert base stations.

Protocol 2 [LOCATE-MULTIPLE]: Strategy of an adversary to locate the positions of multiple hidden base stations

message. Within this time, the MBS moves to position $p_{MBS}(t_2)$. Now, it receives the reply from the node at this position and computes the position of the node based on the time difference with which it received the signal. If this position matches with the position being claimed by the node (allowing for error in computation), then the MBS returns `"Accept"` else it returns `"Reject"` (see figure 10).

I.4 Attack on the protocol in the mobile base stations model

We assume as in [CCS06] that the area within which the positioning of a node takes place is a two-dimensional disk of radius R'. This is a necessary assumption for any reasonable positioning protocol as without such an assumption positioning is trivial, as described in the introduction. For the sake of simplicity, we assume that the location that the adversaries wish to spoof is p_F, the center of this two-dimensional disk. When the position to be spoofed is not the center of the radius, we later provide a lower bound on the probability with which an adversary can spoof a location. Let Δ be the error that a verification authority can tolerate in receiving a signal response from a node. Now, consider a two-dimensional disk of radius R, with center at p_F. The k adversarial nodes $\{A_1, A_2, ..., A_k\}$ will position themselves around the circumference of this two-dimensional disk. We assume that R is small enough such that there is a reasonable probability that the mobile base stations will not enter this two-dimensional disk. These nodes will use directional antennas and broadcast their signals with an angle α as shown in Figure 11. Let the sector in which A_i broadcasts be L_i, A_i, R_i and let radius L_iA_i be equal to a. Let $L_i = R_{i-1}$ for all $2 \leq i \leq k$ and let $L_1 = R_k$. Now, let the angle between A_i, p_F and L_i be θ'.

When responding to a challenge, all the adversaries will delay their response by a time duration of $\frac{R}{s_{us}}$, where s_{us} is the speed of signal transmission. Now, if the angle α is "small enough", then any mobile base station that is in the sector $L_iA_iR_i$ will accept the position p_F due to the signal it received from A_i.

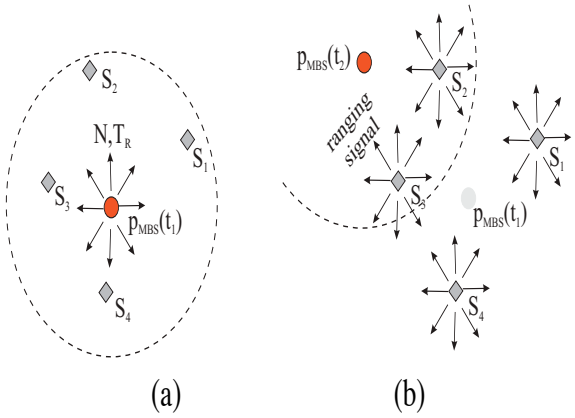

Figure 10: Positioning based on mobile base stations [CCS06]

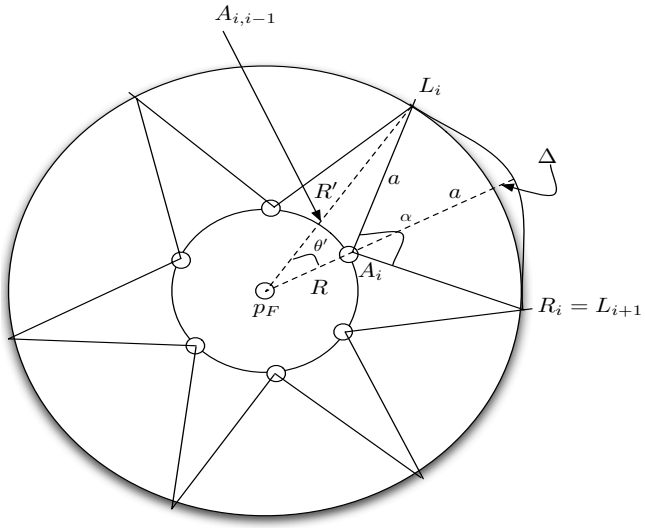

Figure 11: Attack on the protocol in the mobile base stations model

A mobile station that is present in the area $A_iL_iA_{i-1}$ will not receive any signal. For the attack to be successful, it must be the case that no mobile base station is present in such an area. As in [CCS06], we assume that the mobile base station moves uniformly at random over the entire disk of radius R'. The probability (λ) with which the adversaries can falsely prove to a single MBS that there exists a node at p_F, is the ratio of the area $L_iA_iR_i$ to the total area $L_iA_{i-1}A_iR_i$. Thus the probability that the adversaries can make the infrastructure falsely believe that there exists a node at position p_F is λ^{m_b}, where m_b is the total number of mobile base stations in the system. We show later that for reasonable settings of Δ, R and R' this attack is quite reasonable.

Determining k and θ' We now describe how to determine the number of adversaries (k) needed so that θ' will be such that $a + R - R' \leq \Delta$. Note that if an MBS in area $L_iA_iR_i$ were to be present on the line p_fA_i extended to meet the circumference of the disk, then the adversary could spoof location p_F by simply delaying his signal. In the case that MBS is not on this line, then the delaying of signal will cause a slight error in the time at which MBS receives the response signal. We note that fixing θ' gives us a value for a, which in turn will give us a bound on this error. We shall fix an upper bound (θ) on the value of θ' such that the bound on the error is Δ. Once the value of θ is fixed, we can compute the number of adversaries needed. Using properties of triangles we get that,

$$\theta \leq \cos^{-1}\left(\frac{(\Delta + R' - R)^2 - R'^2 - R^2}{-2R'R}\right).$$

Thus to determine how many adversaries we actually need on the circumference, we calculate $k = \lceil \frac{\pi}{\theta} \rceil$. From this, we can determine $\theta' = \frac{\pi}{k}$.

Determining probability of spoofing position p_F First we will calculate the probability of spoofing position p_F to the infrastructure when there is only one mobile base station. Let "good" area (denoted by r_g) be the area $L_iA_iR_i$. This the area in which if an MBS is present, the adversary succeeds in spoofing p_F. Let "bad" area (denoted by r_b) be the area $L_iR'A_i$. This is the area in which the MBS will reject position p_F. The total area $= r_g + r_b$. $\lambda = \frac{r_g}{r_g + r_b}$, is the probability with which an adversary spoofs location p_F to a single MBS. Let $\frac{R'}{R} = \beta$. Now we can calculate the ratio of the areas by simple geometry as

$$\lambda = 1 - \frac{(R' - R)\sqrt{2R^2 - 2R^2\cos(\theta')}\sin(\pi/2 + \theta'/2)}{(R'^2 - R^2)\theta'}$$

$$-\frac{-R^2(\theta' - \sin\theta')}{(R'^2 - R^2)\theta'} = \frac{\beta - \theta}{(1 - \frac{1}{\beta})\theta}$$

When there are m_b mobile base stations in the network, the probability of a successful attack[5] is simply $(\lambda)^{m_b}$.

Spoofing other locations We discussed above, the case where p_F was the center of the disk with radius R'. We outline here, the attack in the case when p_F is some other point on the disk. As before, all adversaries on the circumference of the disk of radius R, will broadcast signals such that the sectors in

[5]More precisely, the success probability is Pr[X]$(\lambda)^{m_b}$ where X is the event that mobile base stations do not enter the disc of radius R. The term Pr[X] is ignored to simplify analysis.

which they broadcast meet exactly on the circumference of disk with radius R'. Now consider the largest disk of radius R_c that can be drawn completely inside the disk of radius R'. Suppose, the positioning were to be done within this new disk and the adversaries broadcast their signals such that the sectors met exactly on the circumference of this new disk. Then, the analysis from above holds and the adversaries will have the probability of success λ (corresponding to $R = R$ and $R' = R_c$) as described earlier. We note that, keeping the number of adversaries the same, if the adversaries were to now instead broadcast such that the sectors met on the circumference of the disk with radius R', then we only increase the probability of success. The reason is as follows: since the distance between two adversaries A_{i-1} and A_i is the same in both cases, the angle $A_{i-1}L_iA_i$ will be smaller in the case where the adversaries broadcast sectors that meet on the circumference of disk with radius R'. Since, this angle determines the area $A_{i-1}L_iA_i$, or in other words r_b, when this angle becomes smaller, r_b will become smaller in comparison with the area $L_iA_iR_i$. Hence, the probability of success of the adversaries will only be larger. Thus, the probability of success of the adversaries in spoofing position p_F (where p_F is not the center of the positioning disk) is certainly greater than λ obtained with corresponding values of R and R_c.

I.5 Analysis of the attack on mobile base stations

We would like to first highlight the following cases. Assume that $R' = 500$m, $R = 50$m and $\Delta = 2.5$m. In this case, we get that we need 11 adversaries and the probability of success of these adversaries against a single mobile base station will be approximately .91. Even with 30 mobile base stations within the 500m disk, the adversaries will still have approximately .06 chance of spoofing location p_F. There will need to be 49 hidden mobile base stations within the 500m disk to push the probability below .01. With $R' = 1000$m $R = 50$m and $\Delta = 2.5$m we will need 11 adversaries. Again we get that the probability of success against a single hidden base station is approximately .95. With 30 mobile hidden base stations the probability of success is .215 and 96 mobile base stations will be needed to drop the adversaries probability of success below 0.01. We give these examples as evidence that our attack is feasible in common reasonable situations. We present below some results about the number of adversaries required and the probabilities of success.

Number of Adversaries Required: Since the actual values of the R', R and Δ do not matter and only their ratio to one another, we only present the ratios. We present two graphs in (figures 12 and 13) with different values of ratios between R, R' and Δ.

Probability of Success Now we look at the probability of success in the "worst case". By worst case, we mean that the value of Δ is very small for a fixed R/R' ratio. This in turn means that the value of θ' is very small and we will need a large number of adversaries. Assume that $\theta' = .00001$. Note that this is only for analysis and will give us a value close to the worst case scenario. It would be impractical to actually have a $\theta' = .00001$ (see graph in figure 14).

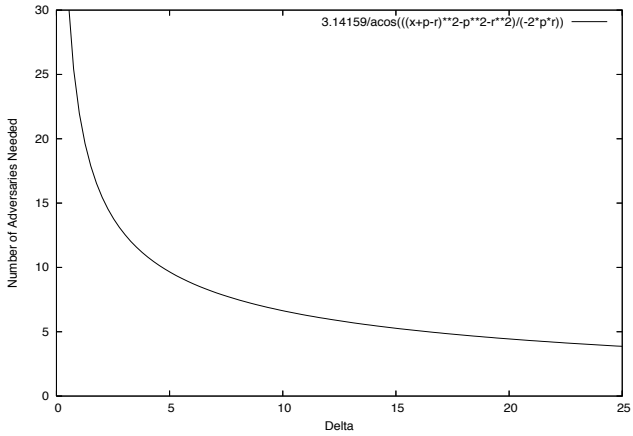

Figure 12: $R = .5R'$ and $\Delta = .01R'$

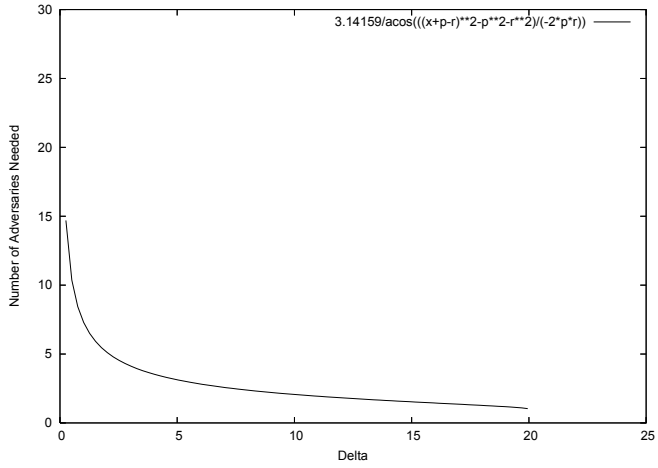

Figure 13: $R = .1R'$ and $\Delta = .01R'$

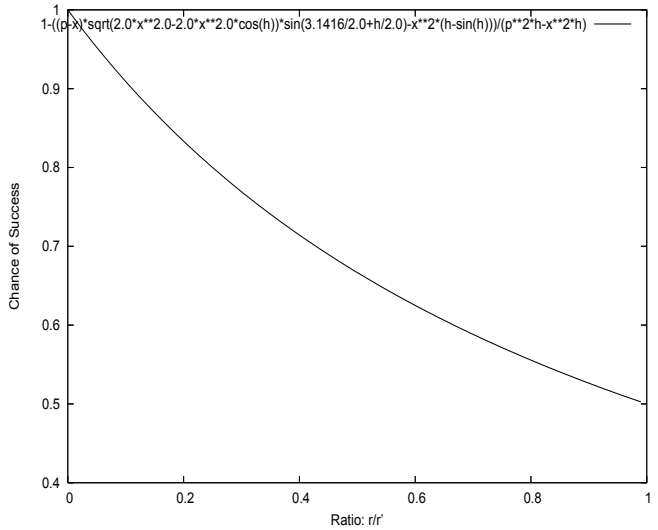

Figure 14: Worst case probability of success