Statistical Model Checking of Distance Fraud Attacks on the Hancke-Kuhn Family of Protocols

Musab A. Alturki
KFUPM, Saudi Arabia
Runtime Verification Inc., USA
musab@kfupm.edu.sa

Max Kanovich
University College London, UK
NRU HSE, Moscow, Russia
m.kanovich@ucl.ac.uk

Tajana Ban Kirigin
Department of Mathematics
University of Rijeka, Croatia
bank@math.uniri.hr

Vivek Nigam
Federal University of Paraíba, Brazil
fortiss, Germany
vivek@ci.ufpb.br

Andre Scedrov
University of Pennsylvania, USA
NRU HSE, Russia
scedrov@math.upenn.edu

Carolyn Talcott
SRI International, USA
clt@csl.sri.com

ABSTRACT

Distance-bounding (DB) protocols protect against relay attacks on proximity-based access control systems. In a DB protocol, the verifier computes an upper bound on the distance to the prover by measuring the time-of-flight of exchanged messages. DB protocols are, however, vulnerable to distance fraud, in which a dishonest prover is able to manipulate the distance bound computed by an honest verifier. Despite their conceptual simplicity, devising a formal characterization of DB protocols and distance fraud attacks that is amenable to automated formal analysis is non-trivial, primarily because of their real-time and probabilistic nature. In this work, we introduce a generic, computational model, based on Rewriting Logic, for formally analyzing various forms of distance fraud, including recently identified timing attacks, on the Hancke-Kuhn family of DB protocols through statistical model checking. While providing an insightful formal characterization on its own, the model enables a practical formal analysis method that can help system designers bridge the gap between conceptual descriptions and low-level designs. In addition to accurately confirming known results, we use the model to define new attack strategies and quantitatively evaluate their effectiveness under realistic assumptions that would otherwise be difficult to reason about manually.

CCS CONCEPTS

• **Security and privacy → Formal methods and theory of security**; **Logic and verification**; **Access control**; • **Computer systems organization → Embedded and cyber-physical systems**; • **Software and its engineering → Model checking**;

KEYWORDS

Distance-bounding protocols, Distance fraud, Probabilistic rewriting, Statistical model checking, Maude

CPS-SPC '18, October 19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5992-4/18/10…$15.00
https://doi.org/10.1145/3264888.3264895

ACM Reference Format:
Musab A. Alturki, Max Kanovich, Tajana Ban Kirigin, Vivek Nigam, Andre Scedrov, and Carolyn Talcott. 2018. Statistical Model Checking of Distance Fraud Attacks on the Hancke-Kuhn Family of Protocols. In *CPS-SPC '18: 2018 Workshop on Cyber-Physical Systems Security and PrivaCy Oct. 19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, Article 4, 12 pages. https://doi.org/10.1145/3264888.3264895

1 INTRODUCTION

Proximity-based access is a typical security requirement in many real-world cyber-physical systems, e.g., smart-card gate-access control systems, contactless payment systems, IoT networks [28], etc. Proximity can also play an important role establishing trust relations needed to authenticate communication in local peer to peer networks. Proximity-based access, however, is well-known to be vulnerable to *relay* attacks, in which an attacker relays messages between a verifier (e.g., a card reader) and a prover (e.g., an NFC tag), maliciously extending the effective range of communication. To protect against relay attacks, distance-bounding (DB) protocols have been introduced [11, 17].

In a DB protocol, the verifier computes an upper bound on the distance to the prover by measuring the time needed for a signal to travel to the prover and back. Since a signal's velocity cannot exceed the speed of light, an upper bound on the distance to the prover can be computed. The round-trip time is measured through a rapid bit exchange, in which the verifier sends a challenge bit to the prover and the prover responds as quickly as possible with the corresponding response bit. Since relaying challenges and responses during this phase introduces relatively significant delays, an attacker's ability to mount a relay attack is severely limited.

DB protocols are, however, susceptible to distance fraud, in which a dishonest prover is able to manipulate the distance bound computed by an honest verifier to make himself appear closer than he actually is, causing the verifier to falsely accept the prover. Distance fraud attacks are *timing* attacks that are mounted without colluding with any external entity, which renders them particularly dangerous. One category of attacks that can lead to distance fraud is *guessing* attacks, in which the attacker attempts to correctly guess the response to the verifier's challenge [11]. Guessing can potentially be used to send out a response in advance (before the challenge arrives) so that the measured round-trip time is reduced. A second category of distance fraud that has been recently

identified is the *in-between-ticks* (IBT) attack [19], which exploits the computational limitations of a verifier and the differences between the physical time and the discrete time of the verifier's clock. Both attack categories build on how sensitive DB protocols are to timing of events, since the slightest timing manipulations can result in significant errors in computing the distance (1 ns ≅ 15 cm). Although both attack categories apply to the rapid bit exchange phase of almost all DB protocols, the Hancke-Kuhn protocol [17] and its variants (including [7, 21, 26] to name a few) are particularly susceptible. This class of protocols, to which we refer as the *Hancke-Kuhn family of protocols*, denoted $\mathcal{HK}$, is characterized by using a public hash function to generate the shared response strings and having a lightweight verification phase that does not require opening commit messages or verifying signatures.

Although guessing attacks, in particular, were identified when the first DB protocol was proposed [11], little has been done since then to investigate formally and systematically their different strategies and countermeasures. Appropriately timing guessed responses and using different guessing strategies can significantly affect the chances of successfully mounting a guessing attack. Moreover, as is typical of manual analysis, the analytical models developed by hand of the recently identified IBT attacks [19] had to rely on some simplifying assumptions to make the analysis tractable. Furthermore, these and most other formal models are necessarily high-level and non-executable, providing only some theoretical guarantees that may not be realizable in a lower-level design or implementation. Executability is particularly important as DB protocols are notorious for being hard to implement in practice. Manually encoding various attack strategies and investigating their effectiveness while taking into account operational details of the protocol and the threat model can be too labor-intensive and is heavily prone to human error.

In this work, we address these limitations by introducing a formal model based on probabilisitic rewrite theories [3] for formally analyzing various forms of distance fraud attacks on $\mathcal{HK}$, which has proven non-trivial due primarily to their real-time and probabilistic nature, despite their conceptual simplicity. The model is both *timed* and *probabilistic*, so that randomized behaviors, environment uncertainties (such as noise) and timing of events can formally be described and analyzed. Furthermore, the model is *generic* and *versatile*, capturing a variety of possible behaviors and attacker models. The model is also *executable*, enabling quick prototyping of different designs and configurations and facilitating automated analysis. Using Maude [13] and PVeStA [4], efficient formal analysis of *quantitative properties*, including probabilities, can be performed using statistical model checking. This methodology provides an efficient and automatic means of verifying complex systems without having to make simplifying assumptions that are usually needed for full probabilistic analysis to be tractable.

Using this model, we show how to accurately and mechanically confirm known results about simple guessing and IBT attacks that were previously shown through manual (and in many cases non-trivial) analysis. We then generalize some of these results to situations involving different levels of noise and acceptance conditions. Additionally, we use the model to define new guessing-ahead attack strategies that have not been investigated before, and then quantitatively evaluate their effectiveness under realistic assumptions involving noise. Moreover, we investigate the interplay of the two attack categories of guessing and IBT attacks. We show, for instance, that certain realistic guessing-ahead attack strategies can significantly benefit from exploiting the IBT vulnerability and mount a successful attack with probability as high as 60% without being detected by the verifier. In general, the analysis enabled by our model provides deeper insights on how attack strategies compare against each other in realistic settings. Moreover, it generally demonstrates how quantitative evaluation based on formal models of DB protocols can be immensely useful.

Contributions. The main contributions are summarized below:

(1) A formal characterization of $\mathcal{HK}$ protocols that is both generic and computational, enabling automated reasoning about various distance fraud attacks in realistic settings;
(2) An efficient and practical formal analysis method than can help system designers realistically assess distance fraud vulnerabilities and experiment with their designs, bridging the gap between high-level formal descriptions and low-level implementations;
(3) Automated confirmation of known distance fraud vulnerabilities, which were previously shown through manual analysis, often assuming idealized behaviors and environments;
(4) A new taxonomy of guessing-ahead attack strategies and new results about their effectiveness with and without the IBT vulnerability in noisy channels.

The rest of the paper is organized as follows. In Section 2, we describe the original Hancke-Kuhn protocol along with distance fraud attacks, followed by an overview of related work in Section 3. Section 4 generally introduces our rewriting model of $\mathcal{HK}$ protocols. Then, in Section 5, we define the properties to be analyzed. After that, Section 6 discusses statistical model checking results for guessing and guessing-ahead attacks, while Section 7 discusses those results for the IBT vulnerability. The paper concludes with a summary and a discussion of future work in Section 8.

2 DISTANCE-BOUNDING PROTOCOLS

Distance-bounding protocols aim to enhance traditional authentication by providing additional assurance in users proximity. Motivation for such requirements is discussed in detail in [17]. Vulnerabilities such as relay attacks can only be addressed by tight integration into the physical layer of the communication protocol.

The goal of a distance-bounding protocol is to ensure access to some resource to valid provers that are within a specified distance bound, and, at the same time, reject access to provers that are located outside of the distance bound perimeter.

There are typically three phases of a DB protocol, an initialization or a setup phase, during which nonces and/or commitments are calculated and/or exchanged, followed by a distance measurement phase that establishes the physical distance, and a finalizing phase used to check commitments, i.e., confirm identification. The distance measurement phase is the essential part of any DB protocol. Both guessing and in-between-ticks attacks, as the main focus of our analysis, are due to failures in the distance measurement phase.

2.1 The Hancke-Kuhn Protocol

As a DB protocol, the Hancke-Kuhn (*HK*) protocol [17] aims to ensure that the prover, P, is in the vicinity of the verifier, V. This

protocol assumes that the prover and the verifier share a long term secret key, K, and a public hash function, h. In the initial (setup) phase of the protocol the verifier and the prover generate nonces N_V and N_P which are used to calculate a sequence of $2n$ bits using K and h: $R_1^0, \ldots, R_n^0 || R_1^1, \ldots, R_n^1$, $R_i^j \in \{0,1\}$. The setup phase of HK protocol is followed by a series of n single-bit exchanges, defined by the following procedure: To a random challenge bit C_i sent by the verifier in the ith round, the prover instantly replies with either R_i^0, in case $C_i = 0$, or R_i^1, in case $C_i = 1$. At the same time, the prover discards the corresponding other bit, R_i^1 or R_i^0. This way, the prover reveals only half of all the bits derived in the initial phase. For each round, the verifier marks the time when a challenge bit is sent, and the time the response is received.

In the last phase of the protocol, the verifier computes his distance from the prover and checks that the responses are correct. The verifier grants access to the prover if all time tests for bit exchanges are successful, i.e., do not exceed the predefined distance bound, and if all n bits are correctly exchanged. Keeping in mind potential errors, due to e.g., noise, the verifier's decision can be parametrised so that access is granted if the time-test is satisfied in a number of rounds, e.g., in a simple majority of rounds, and if only a number of response bits, k out of n, are correct. Different acceptance criteria are further explored when we describe the rewriting model of DB protocols in Section 4.

2.2 Distance Fraud Attacks

Guessing Attacks. The responder involved in the distance measurement phase of a DB protocol can be an *attacker*, who does not share the relevant secrets, including the secret key K, with the verifier, as well as a dishonest prover, having access to the shared secrets. An attacker with no access to the shared secrets may try to successfully complete a protocol session by *randomly* guessing the correct bit responses to the challenge bits received. A dishonest prover may also use guessing to try to appear closer than he actually is. He does so by guessing challenge bits and sending response bits ahead of time, before receiving the challenge bits. This affects the measured time difference, and hence the relative distance calculated by the verifier. Furthermore, as he knows the $2n$ bit sequence used in the protocol session, the dishonest prover may perform *educated guessing*: before receiving a challenge bit C_i, he may randomly choose between potential response bits R_i^0 and R_i^1. Moreover, in the case when $R_i^0 = R_i^1$, the correct response is a priori known to the prover.

While guessing ahead, the prover needs to maintain synchrony with the verifier's pace of sending out challenges to minimize the chances of being detected. This is important since an unsolicited response is a witness for a guessing-ahead attempt and is used by the verifier to immediately abort the protocol session. To synchronize with the verifier, the prover carefully times a premature response r_{i+1} (for round $i+1$) in relation to the challenge c_i of the preceding round i. Therefore, in an n-round protocol run, the last $n-1$ responses can all be sent out prematurely. The first-round response is not guessed ahead as the prover uses the first challenge to initiate the attack. The diagram in Figure 1 illustrates a possible sequence of challenge-response exchanges with guessing ahead.

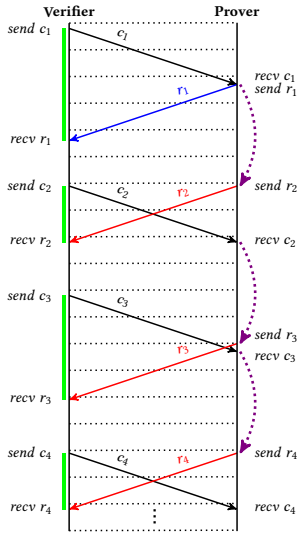


Figure 1: A challenge-response sequence involving a guessing-ahead prover. The dashed horizontal lines mark the verifier's discrete clock cycle boundaries, and the vertical bars on the left highlight the measured time-of-flight.

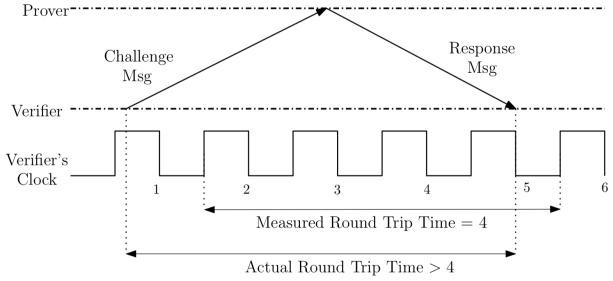


Figure 2: Attack in-between-ticks

In-between-ticks Attack. Another type of attack that exploits the distance measurement phase of a DB protocol is the *in-between-ticks* (IBT) attack identified in [19]. It can appear even when the prover is honest and adversary is not present. It does not involve guessing of the response bits and does not rely on the design details of a specific distance bounding protocol. This attack is a consequence of the foundational difference between real-time in the physical world and time management by discrete time processors that are used as verifiers. Namely, such a discrete time verifier performs instructions and measures time following his clock cycle rate and performance limitations. Assuming that an instruction can be executed in one clock cycle, after sending a challenge message at some moment, the verifier can record this sending time only at a later moment, in another clock cycle. Similarly, when receiving a response message, the verifier records its arrival at a later moment. This can result in discrepancy between the actual and the observable time intervals, i.e., between actual time when the bits are sent and received, on one hand, and the recorded time of sending and receiving bits, on the other, as shown in Figure 2. Consequently,

the verifier can make the erroneous decision to grant access to a prover that is outside the specified perimeter. This error can easily amount to several meters when using radio-frequency and standard devices with not very powerful processors. Moreover, it has also been shown in [19] that the probability of such false acceptance is rather high (up to 50%) when the prover is close to the bound. For more details on this type of attack, including its probability analysis, see [18, 19].

3 RELATED WORK

The approach followed in this work, which is based on Rewriting Logic and Maude, has been used before to formally verify quantitative properties of probabilistic systems, including analysis of resilience against DoS attacks [2, 5], analysis and redesign of wireless sensor networks [20], evaluation of design alternatives of distributed transaction systems [23], among several others. In [15], SeVen, a selective strategy for low-Rate application level DDoS attacks is formalized in Maude and formally analyzed for a range of parameters to gain insight into the potential effectiveness of SeVen.

Probabilistic reasoning in the analysis of cryptographic protocols in general has been supported by tools such as e.g. EasyCrypt [16].

A formal model of physical security protocols that extends the Dolev-Yao model with dense time, network topology, and node location is presented in [8, 29]. The model is formalized in Isabelle/HOL and used to verify distance bounding protocols where the concrete message theory includes exclusive-or. In [9] the model is used for additional protocols including formally analyzing ranging, distance bounding, and secure time synchronization.

In [27], a probabilistic model of guessing, needed to analyze protocols that mix weak cryptography with physical properties of nonstandard communication channels refines the Dolev-Yao algebraic method for protocol analysis. The model is used to develop a precise security proof for *HK* protocol

Attacks that can be found by models of cyber-physical security protocols using dense time, but not when using discrete time are presented in [19]. This is illustrated with the IBT attack, which can be carried out on most DB protocols. A probabilistic analysis of the attack is also presented.

A unified framework is introduced in [6] that aims to improve analysis and design of DB protocols. The framework characterizes different attacks, adversary/prover capabilities and strategies. It introduces notions of black-box and white-box models, and the relation between the different attacks with respect to these models. The framework is demonstrated with a detailed analysis of the Munilla-Peinado DB protocol. The framework is not formalized and analysis is carried out by hand.

An exhaustive classification for attacks on DB protocols is defined in [14] that includes a new attack called Distance Hijacking Attack. Countermeasures for several attacks are proposed. The formal framework of [8, 29] is extended to support reasoning about overshadowing attacks and the resulting framework is used to prove the absence of attacks after the proposed countermeasures are applied. However, all considered attacks are caused by failures in the authentication phase of DB protocols, not by failures of the distance measurement phase of DB protocols that we consider in our analysis here.

4 A REWRITING MODEL OF $\mathcal{HK}$

We use Rewriting Logic [24], and its probabilistic extensions [3, 22], to build a generic and executable model of $\mathcal{HK}$ protocols. The model captures at a high level of abstraction the essential structural and behavioral aspects of a $\mathcal{HK}$ protocol. Furthermore, using Maude [13], a high-performance Rewriting Logic engine, the model can be simulated to generate random sample executions that can be used to statistically model-check probabilistic properties of the protocol, including resilience against distance fraud attacks. In this section, we quickly review Rewriting Logic and Maude and then give a general description of the model and its behaviors.

4.1 Rewrite Theories and Maude

A *rewrite theory* $\mathcal{R}$ formally describes a concurrent system including its static structure and dynamic behavior. It is a tuple $(\Sigma, E \cup A, R)$ consisting of: (1) a membership equational logic (MEL) [25] signature Σ that declares the kinds, sorts and operators to be used in the specification; (2) a set E of Σ-sentences, which are universally quantified Horn clauses with atoms that are either equations $(t = t')$ or memberships $(t : s)$ (where t and t' are terms and s is a sort); (3) a set A of equational axioms, such as commutativity, associativity and/or identity axioms; and (4) a set R of rewrite rules $t \longrightarrow t'$ if C specifying the computational behavior of the system. (See [12] for a detailed account of generalized rewrite theories).

Probabilistic rewrite theories extend regular rewrite theories with probabilistic rules [3, 22]. A probabilistic rule

$$t \longrightarrow t' \text{ if } C \text{ with probability } \pi$$

specifies a transition that can be taken with a probability that may depend on a probability distribution function π parametrized by a t-matching substitution satisfying the condition C. Probabilistic rewrite theories unify many different probabilistic models and can express systems involving both probabilistic and nondeterministic features. A more detailed account of probabilistic rewrite theories can be found in [22, 30].

Probabilistic rewrite theories, specified as *system modules* in Maude [13], can be simulated by sampling from probability distributions. Using PVeStA [4], randomized simulations generated in this fashion can be used to statistically model check quantitative properties of the system. These properties are specified in a rich, quantitative temporal logic, QuaTEx [3], in which real-valued state and path functions are used instead of boolean state and path predicates to quantitatively specify properties about probabilistic models. QuaTEx supports parameterized recursive function declarations, a standard conditional construct, and a *next* modal operator $\bigcirc$, allowing for an expressive language for real-valued temporal properties (Example QuaTEx expressions appear in Section 5). Given a QuaTEx path expression and a Maude module specifying a probabilistic rewrite theory, statistical quantitative analysis is performed by estimating the *expected value* of the path expression against computation paths obtained by Monte Carlo simulations. More details can be found in [3].

4.2 An Overview of the $\mathcal{HK}$ Model

We introduce a model of $\mathcal{HK}$ as a probabilistic rewrite theory $\mathcal{R}_{\mathcal{HK}} = (\Sigma_{\mathcal{HK}}, E_{\mathcal{HK}} \cup A_{\mathcal{HK}}, R_{\mathcal{HK}})$. The full specification of

the model in Maude is available online at https://bitbucket.org/malturki/dbp2/. Since our aim is to be able to formally model and analyze distance fraud attacks, for which the rapid bit exchange phase is the most relevant phase of a DB protocol, the other two phases (the setup and verification phases) are only abstractly specified. This keeps the model generic, enabling reasoning about distance fraud attacks in different $\mathcal{HK}$ protocols. Furthermore, the model is heavily parametrized to capture different attack behaviors and countermeasures, further adding to its generic design. Moreover, by utilizing different facilities provided by its underlying formalism, the model is both probabilistic, specifying randomized behaviors and environment uncertainties, and real-time, capturing time clocks and message transmission delays.

The structure of the model, specified by the MEL sub-theory $(\Sigma_{\mathcal{HK}}, E_{\mathcal{HK}} \cup A_{\mathcal{HK}})$, is based on a representation of actors in Rewriting Logic, which builds on its object-based modeling facilities. In this model, the state of the protocol is a *configuration* consisting of a multiset of actor objects and messages. Objects communicate asynchronously by message passing. There are two fundamental objects in the model: the verifier object and the prover object.

The verifier object maintains in its state a `status` attribute indicating its current execution step in the protocol, and whether the current protocol run has successfully or abnormally terminated. Furthermore, the verifier maintains a `lists` attribute containing the bit strings that are generated during the setup phase and used during the rapid bit exchange phase. In addition, the values of both the selected challenge bit and the received response bit for the current round are recorded by the verifier. Moreover, to measure the time needed for a challenge-response round to complete, the verifier has to record the measured time value for when the challenge was sent and the measured time value for when the corresponding response was received. The difference between these two values gives the *measured* round-trip time, and hence the measured distance. Furthermore, to enable modeling absence of IBT attacks, we also include two attributes for recording the *actual* time values for sending a challenge and receiving the corresponding response. The difference gives the *actual* round-trip time. While the measured time is what a realistic verifier having limited processing capabilities would compute, the actual time is computable by an abstract verifier with an extremely powerful and precise processor that is not vulnerable to the IBT attack (see [18]). Finally, the verifier keeps track of: (1) the number of rounds the response's measured round-trip time was within the bound, (2) the number of rounds the response's actual round-trip time was within the bound, and (3) the number of rounds the response bit was correct. The desired acceptance criteria can later be defined as a function of one or more of these counters (see Section 5).

The prover object is a simpler object maintaining a `status` attribute, a `lists` attribute, and an internal counter for counting incoming challenges.

The model uses dense time (represented by real numbers) to model physical timing of events. Moreover, the (implicit) discrete clock of the verifier object is assumed to always be in sync with the dense physical time of the configuration. In other words, assuming that both physical time and the verifier's clock begin at 0, discrete clock ticks will occur exactly at the positive-integer-valued instants of time.

4.3 $\mathcal{HK}$ Model Parameters

The model is designed to be parametric in a number of variables that can be adjusted as needed for the analysis task at hand. The parameters are specified as operators in $\Sigma_{\mathcal{HK}}$ and instantiated using equations in $E_{\mathcal{HK}}$.

Protocol parameters. Security parameters of the protocol include the number of rounds of rapid-bit exchange, ROUNDS, and the protocol's distance upper bound (measured in time delay), given by MAXRTT. Furthermore, BBASE and DEPTH define, respectively, the number of possible unique bit values and the number of response bit strings to be generated by the protocol's hash function. These two parameters enable modeling generalized versions of the *HK* protocol with k-valued challenges and responses across m strings, with $k, m \geq 2$. Finally, noise levels in the communication medium are captured by the noise bias parameter, NOISE $\in [0, 1]$, which is the probability of a bit being destroyed while in transit due to noise. Note that, unlike *HK* [17], where noise is assumed to simply *flip* bits (which is rather unrealistic), we model noise as (random) disturbance from which the original signal (bit) cannot be recovered.

Threat model. Three parameters capture time delays pertaining to actions made by the verifier: X, the time delay to send out a challenge, and Y and Z, the time delays to record timestamps of a challenge sent and a response received, respectively. The IBT vulnerability can be enabled by setting the Boolean flag IBT? (for a realistic verifier) or disabled by resetting it to false (for an abstract verifier with access to physical timestamps). Furthermore, the behavior of the prover is defined by the parameter PTYPE, which can take one of three values: 0 for a legitimate prover (no guessing), 1 for a dishonest prover that has access to the protocol session's bit strings (educated guessing), and 2 for an attacker that may not have access to the protocol session's bit strings (random guessing). In the cases of a dishonest prover (when PTYPE > 0), whether the prover is able to mount a guess-ahead attack is determined by the Boolean flag GAHEAD?, and in this case, the parameter GATD specifies how much in advance the attacker/dishonest prover is sending out his premature responses to the verifier. Finally, an upper bound on the (actual) distance between the verifier and the prover is determined by the sum MAXRTT+H, where H is the distance differential, which can in principle have a value in the range $(-\text{MAXRTT}, \infty)$. While an attacker will have a positive value for H (representing being outside the verifier's bound), a legitimate prover is modeled by having H ≤ 0.

Acceptance threshold. Three parameters define acceptance threshold levels for a run of the protocol, one for each acceptance criterion. The parameters are: (1) MIN-MTR, the minimum number of successful rounds based on the measured round-trip time, (2) MIN-ATR, based on the actual round-trip time, and (3) MIN-MBR, based on the correctness of the response bits. Values may range from 0 up to ROUNDS (which represents the most restrictive acceptance condition requiring all rounds to be successful). Of course, simple majority and large majority can be defined as functions on ROUNDS, and final acceptance can be based on a combination of these parameters (e.g., a large majority for MIN-MTR and a simple majority for MIN-MBR).

We note that using these parameters, different models can be obtained as special instances. For example, the simple model of the *HK* protocol is obtained by assuming binary bits with two bit sequences (BBASE = DEPTH = 2), absence of the IBT vulnerability (IBT? = false), an attacker guessing based on the shared bit sequences (PTYPE = 1), no guess-ahead attacks (GAHEAD? = false), a positive number of rounds (ROUNDS > 1), acceptance based on both response correctness and measured distance (with MIN-MBR = MIN-MTR = ROUNDS), and, finally, no noise (NOISE = 0).

4.4 $\mathcal{HK}$ Model Transitions

The behaviors of the verifier and the prover in the protocol are specified by, possibly conditional and probabilistic, rewrite rules R_{db} in $\mathcal{R}_{db}$. There are ten, semantically rich rewrite rules in R_{db} in total (of which six rules are probabilistic). To save space and avoid presentation clutter, we describe the main events associated with the rewrite rules specifying the model's bit exchange behavior instead of including the rules verbatim from the specification. As mentioned above, more details can be found at https://bitbucket.org/malturki/dbp2/.

Initiating a round. The beginning of a round is triggered by the verifier object receiving a self-addressed `beginRound` message while in the `ready` state (self-addressed messages are commonly used in actor-based systems to schedule internal events [1]). When starting a round, the verifier prepares itself by resetting any stored time value from the previous round and randomly selecting the challenge bit to be used for this round from the set {0, ..., BBASE−1}. Furthermore, the verifier goes into a `sending` state and schedules a self-addressed `sendChallenge` message to send out the selected challenge.

Sending a challenge. When it is time to send out the challenge, the verifier schedules two messages: (1) a self-addressed message `recordTime` scheduled to be delivered at the next discrete clock tick for the verifier to timestamp the sending of the challenge, and (2) a `challenge` message having the challenge bit previously selected by the verifier, addressed to the prover and scheduled for delivery with a delay equal to the sum of X (the verifier's internal delay) and the transmission delay (computed as half the sum MAXRTT + H). Furthermore, the actual time for sending the challenge is recorded. Finally, the verifier changes its status to `sent`.

Recording the timestamp. When the message `recordTime` arrives, the verifier records the timestamp t + Y, where t is the current time value, and Y is the random variable associated with recording a timestamp for sending the challenge. The verifier goes into a `receiving` state and no new messages are scheduled.

Responding to a challenge. Upon receiving a challenge from the verifier, the prover reacts based on its type and the current state of the bit exchange phase. If the protocol run consists of only one round of bit exchange (ROUNDS is 1) or the prover is not attempting to guess-ahead responses (GAHEAD? is false), the prover simply schedules a 'standard' response to the challenge received to be delivered after a delay given by the one-way transmission delay d. Otherwise, if the bit exchange is multi-round *and* the prover is guessing ahead, there are three distinguishable cases:

(1) The current round is the first: the prover schedules two responses: (1) a standard response to the challenge received delayed by d, and (2) a guess-ahead response in anticipation of the next challenge (the second challenge) delayed by $3d$ − GATD, which is the amount of time the prover anticipates the verifier to expect the next response, minus the guess-ahead time differential GATD.
(2) The current round is neither the first nor the last: the prover does not respond to the received challenge (as it had already responded with a guess in the previous round). Instead, the prover schedules only a guess-ahead response for the next challenge.
(3) The current round is the last: the prover does not schedule any response, since all the responses needed have already been scheduled before.

The payload of the response message is equationally computed and may depend on the value of the challenge bit received, the current-round and/or next-round tuples of bits in the shared bit strings, the type of the prover PTYPE, and the noise level NOISE. In any case, if noise succeeds, the response is simply turned into an illegible signal.

Receiving the response. When the prover's response message arrives, and the verifier is in the `receiving` state (implying that the verifier is expecting a response), the verifier records the actual time the response was received, records the response bit, changes its status to `received`, and schedules a self-addressed message to be delivered by the next discrete clock tick with a time delay given by Z (the random delay associated with recording the timestamp of the response). If the prover's response message arrives while the verifier is not expecting a response (indicated by a status field value different from `receiving`), which can only happen if the prover is mounting a guess-ahead attack, the verifier detects the attack and immediately aborts the protocol session (the status is set to `aborted`).

Recording the timestamp. In this last step of a bit-exchange round, the verifier records its measured timestamp of receiving the response, appropriately updates its counters keeping track of successful rounds, and goes into the `ready` state in preparation for the next round (if any).

5 MODEL CHECKING DISTANCE FRAUD

Two fundamental security properties of a DB protocol are *false acceptance* and *false rejection* probabilities. False acceptance (FA) occurs when an honest or a dishonest prover outside the allowed perimeter or an attacker is granted access to the protected resource by the (honest) verifier, signaling a successful attack. False rejection (FR), on the other hand, happens when an honest prover is unrightfully denied access to the resource by the (honest) verifier, despite being close enough to the verifier.

To formally analyze FA and FR using the rewriting model of $\mathcal{R}_{\mathcal{HK}}$, we first express acceptance and rejection in a DB protocol as temporal formulas in QuaTEx. The acceptance formula has the following form:

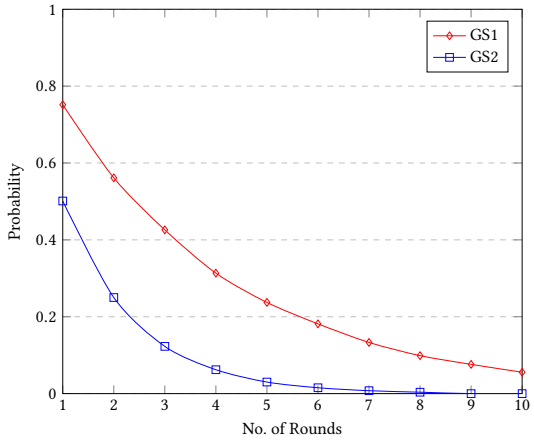
$$\begin{aligned} accepted(t) = \ & \underline{\textbf{if}}\ time() > t\ \underline{\textbf{then}}\ hasSatisfiedThreshold() \\ & \underline{\textbf{else}}\ \bigcirc\, accepted(t)\ \underline{\textbf{fi}}\ ; \\ & \underline{\textbf{eval}}\ \mathbf{E}[accepted(t_0)] \end{aligned}$$

The parameter t is the time limit beyond which protocol execution is halted. In $\mathcal{R}_{\mathcal{HK}}$, the limit (given by the actual parameter t_0) is set so that the DB protocol is guaranteed to execute all the way to completion (or until the verifier aborts execution). *accepted*(t) is a recursively defined path expression that uses two state functions: (1) *time*(), which evaluates to the time value of the current state of the protocol, and (2) *hasSatisfiedThreshold*(), which evaluates to 1.0 if the current state of the verifier object indicates that the minimum acceptance threshold has been met, and 0.0 otherwise. Therefore, given an execution path, the path expression *accepted*(t) evaluates to *hasSatisfiedThreshold*() in the current state if the protocol run is complete; otherwise, it returns the result of evaluating itself in the next state, denoted by the next-state temporal operator ◯. The probability of acceptance can be approximated by estimating the expected value of the formula over random runs of the protocol, denoted by the query **eval** E[*accepted*(t_0)]. Different acceptance criteria, e.g. measured-time acceptance and bit-correctness acceptance, give rise to different versions of the formula *accepted*(t).

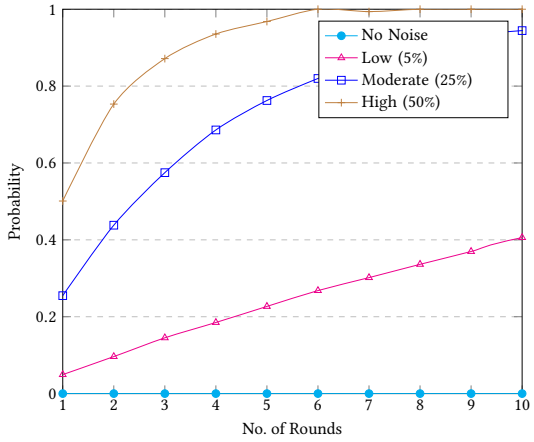
The rejection formula *rejected*(t) is the dual of the acceptance formula and its definition is similar. A statistical approximation of the rejection probability is similarly computed using the query **eval** E[*rejected*(t_0)].

A third security property, which is relevant only in the presence of a guessing-ahead attacker, is the *attack detection* (AD) rate, which is the probability that the verifier receives an unsolicited response. This property is similarly specified as a QuaTEx formula.

We have used our model given by $\mathcal{R}_{\mathcal{HK}}$ to statistically model check FA and FR formulas (and AD when applicable) under various settings using the statistical model checking tool PVeStA. In particular, in the following sections we analyze: (i) *Guessing attacks* (Section 6), including simple guessing and guessing-ahead attacks, and (ii) the *IBT vulnerability* (Section 7), including IBT-only attacks and guessing ahead while exploiting IBT. Throughout the analysis, we assume a 99% confidence interval with size at most 0.02. We also fix MAXRTT in $\mathcal{R}_{\mathcal{HK}}$ to 4.0 time units (similar to [18, 19]) and assume a uniform distribution for the verifier's delays X, Y and Z.



(a) FA probability with guessing



(b) FR probability due to noise

Figure 3: False Acceptance (FA) and False Rejection (FR) probabilities based on bit correctness in multiple rounds

6 GUESSING ATTACKS

$\mathcal{HK}$ protocols are particularly vulnerable to guessing attacks, since the event space determined by the possible bit values is usually small (typically only binary bits are used) and the bit-exchange rounds – in many cases – are probabilistically independent. In our analysis, we consider the two possible guessing behaviors: *simple guessing* and *guessing ahead*, defined by whether the response is sent out after or before a challenge is received.

6.1 Simple Guessing Attacks

Simple guessing targets acceptance based on correctness of responses. Resilience to simple guessing attacks is a good basic measure of a protocol's resilience to more sophisticated attacks. As mentioned before, there are two possible guessing strategies: educated guessing (denoted by GS1), that requires access to protocol session's bit strings, and random guessing (denoted by GS2). In GS1, the probability of FA in a single round is 0.75 [17], while for GS2, the probability is only 0.5, which is the optimal FA probability of a DB protocol. To reduce the probability of FA, the bit-exchange phase is normally run in $n > 1$ rounds, so that for GS1 and GS2, the probabilities are reduced to 0.75^n and 0.5^n, respectively. These results are easily confirmed by our model for single-round and multi-round runs of the protocol, as the plot in Figure 3(a) shows.

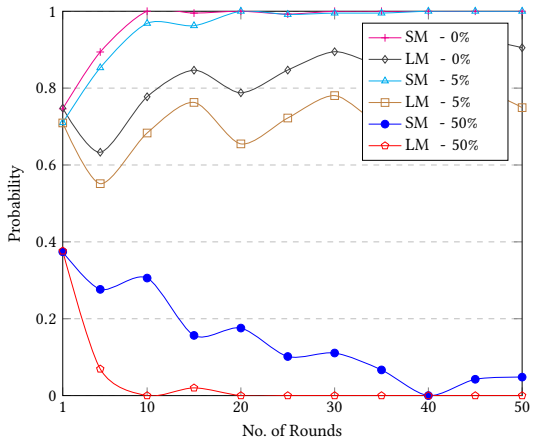
In the presence of noise, to which ultra-wide band communication channels commonly targeted for DB applications are very susceptible, a legitimate prover may be denied access causing FR. To investigate the effects of noise, we estimate the probability of FR assuming three levels of noise: low (a 5% chance of a bit being destroyed due to noise), moderate (25%) and high (50%). Figure 3(b) plots the estimated probabilities against the number of rounds used. As noise levels increase, the probability of FR increases significantly, especially for larger numbers of rounds. Figure 3 demonstrates that reducing FA and FR are two conflicting requirements when deciding the number of rounds to be used with noisy channels.

One way to reduce such large FR rates is to loosen the acceptance condition so that a prover is accepted whenever he succeeds in at least $k < n$ rounds (set by the parameter MIN-MBR), where n is the number of rounds. We investigate two acceptance thresholds:

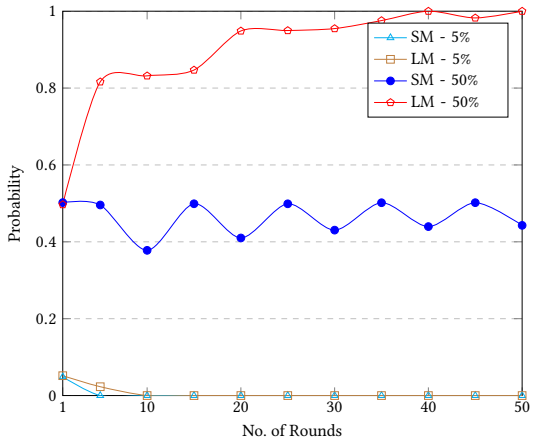
the simple majority (SM), where $2k \geq n$, and large majority (LM), where $3k \geq 2n$. Figure 4(a) plots FA probabilities assuming a GS1 attacker, while Figure 4(b) plots FR probabilities assuming an honest prover, all at low (5%) and high (50%) noise levels (the zero-noise cases in Figure 4(a) are shown for comparison only).

We first point out the wavy lines, which are characteristic of SM and LM data points and show up in all charts involving SM and LM measurements. This phenomenon is due to the fact that the minimum threshold values for SM and LM acceptance may represent percentages higher than 50% and 67%, respectively, of the total number of rounds, affecting FA and FR. For example, in SM, when ROUNDS is 5, the minimum threshold is 3 rounds, representing a 60% acceptance requirement and resulting in a lower FA rate. As the number of rounds increases, however, this phenomenon diminishes as the computed thresholds converge to their target percentages.

More importantly, we observe from Figure 4(a) that noise does not help simple guessing attacks. In fact, the higher the noise, the less effective the attacks are. Indeed, for an attack round to be successful, the prover must make the correct guess *and* noise will have to fail. The effect of noise is especially prominent when noise is high for large values of ROUNDS, where FA is almost zero for both SM and LM. We also note that for low noise at 5% (and similarly at 0%), FA probabilities are quite high, in the range $0.55 - 1.0$. Although this may seem counter-intuitive at first thought, it should be expected since the acceptance requirements of SM and LM (50% and 67%, respectively) are lower than the success probability of a GS1 prover, which is 75%. Nevertheless, LM performs consistently better than SM in reducing FA across all noise levels. Unfortunately, this comes at the price of increased probability of denying access to legitimate users, as can be seen by comparing the plots in Figure 4a and 4b. SM, however, seems to provide the best compromise for very high levels of noise.



(a) FA probability (assuming GS1)



(b) FR probability

Figure 4: False Acceptance (FA) and False Rejection (FR) probabilities for simple majority (SM) and large majority (LM) for bit correctness

6.2 Multi-valued Challenges and Responses

In addition to the number of rounds, two more security parameters can be adjusted to reduce FA. Although they have been classically fixed at 2, the number of distinct challenge bit values, and hence the number of bit strings generated by the shared hash function, and the number of distinct response bit values, can each be increased beyond 2. k-valued challenges and m-valued responses, with $k, m > 2$, were explicitly considered in the SKI protocol scheme of [10]. Higher values of k and m can significantly strengthen an $\mathcal{HK}$ against FA. More sophisticated hardware components and signaling techniques that are needed to cope with the increased complexity exist and can be employed for realizations of such designs.

Our model captures multi-valued challenges and responses using the parameters DEPTH and BBASE, respectively. Figure 5 plots FA against the number of rounds for different DEPTH x BBASE pair values. Clearly, FA probability drops rapidly as the DEPTH and BBASE values are increased. In particular, 4-valued bits already surpass the optimal performance of a standard DB protocol even with a GS1 prover with access to the shared strings. Using different methodology, we, therefore, present an additional argument in favour of multi-valued bit approach considered in [10].

A variation of $\mathcal{HK}$[1] aimed at reducing FA to an optimal 0.5^n is based on the use of fully complementary sequences of bits, so that each pair of strings has the maximum Hamming distance, equal to the number of rounds. Obviously, such complementary sequences require that BBASE $\geq$ DEPTH. Using the idea of the complement, as the analysis in Figure 5 shows, educated guessing becomes only as effective as random guessing, as expected. It is important to note, however, that these results hold for distance fraud only, which is the focus of our analysis. Other types of fraud, involving a third-party intruder (different from the prover), may actually benefit from this setup and increase FA. Namely, the exchanged bits may enable the external attacker to gain some knowledge about the structure of the bit strings that he can then exploit (The worst case occurs when binary bits are used, since the attacker can fully construct the two strings by observing the bit exchange between the unsuspecting verifier and prover).

[1]Suggested by Joshua D. Guttman through personal communication during the Protocol eXchange meeting on December 12, 2017

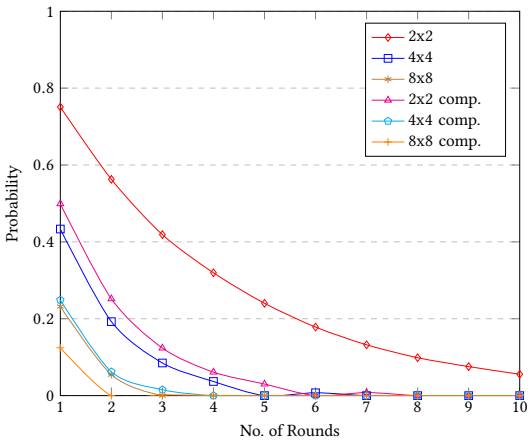


Figure 5: False Acceptance (FA) probability with multi-valued bits and multiple strings (for bit correctness in all rounds)

Attack Strategy		Aim/Assumptions	Guess-ahead
Prescient PR_h		Verifier's location is known exactly	Exactly $2h$[†]
Conservative	CN_h	Avoid detection with uncertainty in verifier's exact location	Chosen uniformly at random from $[h, 2h]$
	CN	Avoid detection with no knowledge of verifier's location	Chosen uniformly at random from $[T/4, T/2]$[‡]
Aggressive	AG_h	Meet the time bound with uncertainty in verifier's exact location	Chosen uniformly at random from $[3h/2, 5h/2]$
	AG	Meet the time bound with no knowledge of verifier's location	Chosen uniformly at random from $[T/2, T]$

[†] h is the prover's distance from the bound.
[‡] T is the verifier's clock cycle period.

Table 1: A taxonomy of guessing-ahead strategies

6.3 Guessing-ahead Attacks

Guessing-ahead attacks target acceptance based on both correctness of the response bits and the calculated time bounds. Consequently, they represent more complicated scenarios than the ones explored earlier in this section.

In the simplistic, and rather unrealistic, case that a verifier's processing delay between receiving a response and sending out the next-round challenge is *fixed*, say d, a guessing-ahead prover can simply add this delay d to his guess-ahead window to ensure that the attack goes undetected. Even if d is initially unknown, the attacker may use the first two rounds to learn d and then take it into account in subsequent rounds, since it is fixed. However, in reality, d is not fixed as it depends on the verifier's state when the response is received along with the (unpredictable) verifier's hardware delays associated with time-stamping and sending messages. Variability of d is captured to some extent in our rewriting model by having the verifier's actions governed by a discrete clock and by modeling the (bounded) random time delay X for sending out challenges. In general, randomizing d across rounds can potentially serve as a countermeasure for guess-ahead attacks [11]. It would therefore be of interest to analyze guessing-ahead attacks in the presence of such randomized verifier behavior.

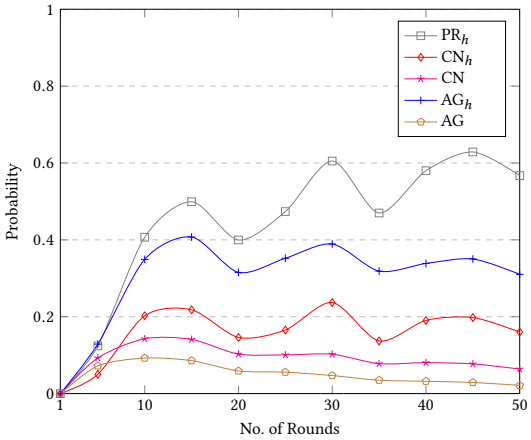
While the distance bound for a DB protocol is usually public knowledge (given as part of the specification of the protocol), the exact location of the verifier may not necessarily be known to the prover. This means that a prover attempting a guess-ahead attack may not know exactly what his distance h from the bound is, although he may still be aware of the verifier's approximate location. Therefore, we define the following guess-ahead strategies:

(1) The *prescient strategy* (PR_h), which represents a prover who knows the verifier's exact location (and hence h) and, therefore, responds exactly $2h$ time units sooner than the anticipated time of receiving the challenge. It models a strong attacker and serves as a benchmark for the other attack strategies' performance.
(2) The *conservative strategy* (CN_h), representing a prover who is uncertain about the exact location of the verifier and who aims to minimize the probability of the attack being detected. In this strategy, the guess-ahead time is chosen uniformly at random from $[h, 2h]$. A variation of this strategy, denoted CN, draws its guess-ahead time from the window $[T/4, T/2]$, where T is the verifier's clock cycle period. CN can be used if the uncertainty about the verifiers' location, i.e., h, is very high, and yet the prover is conservative in timing the attack.
(3) The *aggressive strategy* (AG_h), uses more aggressive guess-ahead timing (chosen uniformly from $[3h/2, 5h/2]$), hoping to increase his chances of meeting the time bound requirement while risking being detected. As above, the variant AG, in which the guess-ahead time is drawn from $[T/2, T]$, models an aggressive strategy with unknown h.

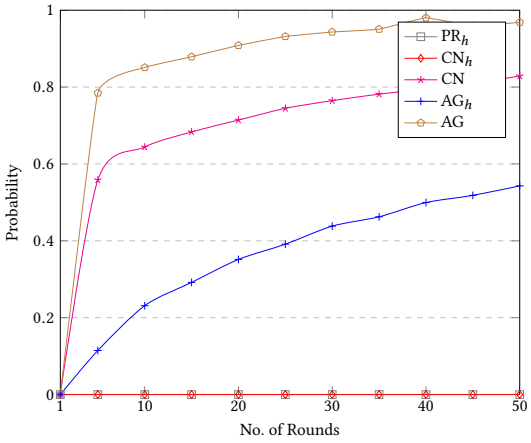
Table 1 summarizes this taxonomy of strategies, which are the ones we consider in our analysis. We show below the results of the most interesting cases, where we have a capable attacker (GS1) who is fairly close to the distance bound ($0 < h \leq 0.5$) and a somewhat cautious verifier using the large majority for both bit correctness and time tests with a typical level of noise (5%). Other possibilities can also be analyzed but they are not expected to yield new or interesting results.

Figure 6(a) plots the probability of a successful attack using these attack strategies for different rounds. We first observe that knowing something about h can significantly increase an attacker's chances of success. The prescient strategy PR_h can achieve success probabilities ranging from 40% to 60%, even at high round counts (and in the presence of noise!). Even if the exact value of h is unknown, the strategy AG_h can achieve 30% to 40% success, which is still quite high, and it outperforms the conservative strategy CN_h. However, knowing very little about the location of the verifier severely limits the effectiveness of the attack (CN and AG). Furthermore, in this case, following an aggressive strategy (AG) is counter-productive.

There are four ways in which a guessing-ahead round may fail: (1) the response bit is incorrect, (2) the response arrives too late, (3) the response gets destroyed by noise, or (4) the response arrives too



(a) FA probability



(b) AD probability

Figure 6: False Acceptance (FA) and Attack Detection (AD) probabilities with guessing-ahead (GS1), low noise and large majority (LM) for both bit correctness and time tests

early (the attack is detected). We have looked into the first three already. It would therefore be insightful to investigate round failures due to the attack being detected by the verifier, i.e., to analyze AD probabilities. Figure 6(b) plots the AD probability for the different attack strategies. While PR$_h$ and CN$_h$ are never detected by the verifier, the AD rates for the other strategies increase with the number of rounds, with the blindly aggressive strategy AG being the worst performer. AG$_h$, however, maintains a detection rate that is comparable to the much more conservative strategy CN, despite being much more effective than CN (as Figure 6(a) shows). By knowing approximately where the verifier is located, an aggressive guessing-ahead strategy can be quite effective.

7 THE IBT VULNERABILITY

In this section, we analyze configurations with an "actual" verifier, which suffers inherent limitations in timely processing instructions against its discrete clock and which does not have access to the physical timestamps of messages sent or received [19]. The IBT vulnerability is enabled in the model by setting the flag IBT? to true. We note, as mentioned before, that this vulnerability can appear with an honest prover and in the absence of an adversary.

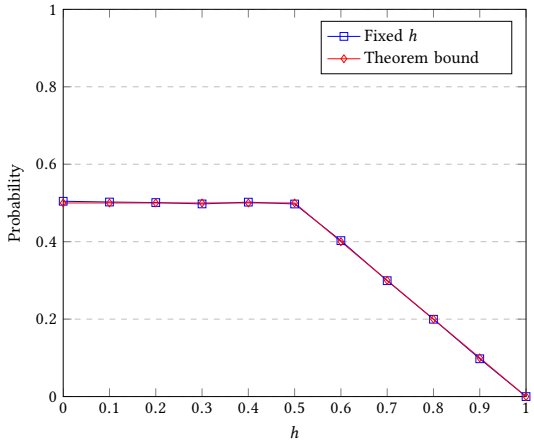


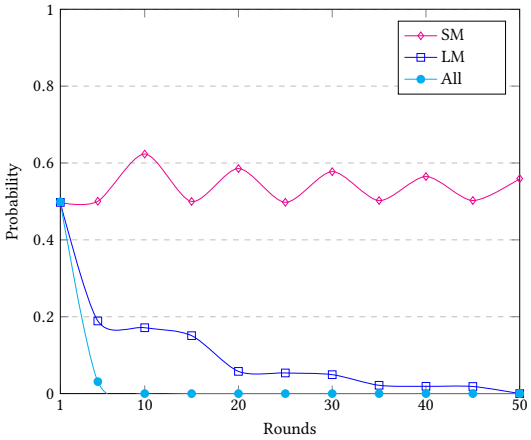
Figure 7: False Acceptance (FA) probability in the (single-round) DB protocol for different values of h (prover's distance from the bound)

7.1 IBT Attacks

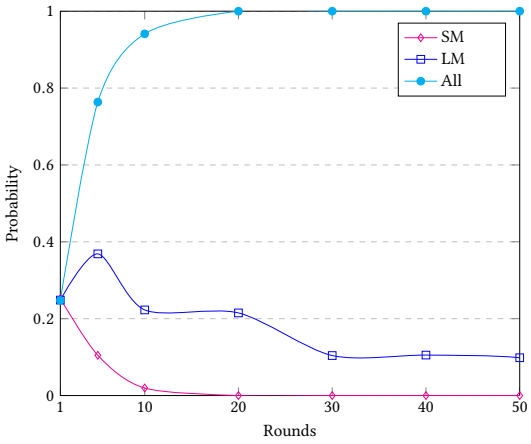
Through the IBT vulnerability, an attacker targets acceptance based on the round-trip time measured by the verifier. To increase his chances of acceptance, the attacker needs to be close enough (but not necessarily too close) to the upper bound stipulated by the protocol (MAXRTT in the model).

We confirm FA probabilities shown in Theorem 4.3 of [19]. The theorem states that the attacker only needs to be within half-a-clock tick from the bound (so $0 < h \leq 0.5$) to succeed with probability 0.5. Moreover, as the attacker moves away from the bound with $0.5 < h < 1$, the probability of mounting an attack linearly decreases as h increases until it becomes zero for $h \geq 1$. Figure 7 plots FA based on the round-trip time measured by the verifier against the bounds given by Theorem 4.3 of [19], using different values of h ranging from 0 to 1. As the chart shows, the estimated probabilities match very closely those given by the theorem. It is worth noting that the analytical models and machinery used for proving this theorem in [19] are non-trivial and achieving such precise estimations mechanically from our model highlights the effectiveness of the model.

Next, we investigate repeating the bit-exchange round as an IBT attack mitigation measure. The simple majority (SM) and large majority (LM) acceptance thresholds are defined as above. In [18], it was shown that SM has no positive or negative effect on FA when $0 < h \leq 0.5$ but can be effective for $0.5 < h < 1.0$, especially as the number of rounds n increases (Theorem IV.7 in [18]). On the other hand, LM decreases FA significantly as the number of rounds n increases for any positive h (Theorem IV.8 in [18]). These results are confirmed by our model while gaining a deeper understanding of how these different thresholds compare. Figure 8(a) plots the probability of FA using both acceptance thresholds when the protocol is run for different values of n. For comparison, we also include



(a) FA probability with $0 < h \leq 0.5$



(b) FR probability with $-0.5 < h \leq 0$

Figure 8: False Acceptance (FA) and False Rejection (FR) probabilities in the multi-round DB protocol for different time test acceptance conditions (SM, LM, All) and prover's distance from the bound (h)

the most restrictive 'All' acceptance threshold requiring all rounds to pass the bound test successfully. Figure 8(a) shows that, for SM, FA remains close to its single-round value (0.5) as n increases, while LM reduces FA considerably. For $0.5 < h < 1.0$ (not shown in the chart), both acceptance thresholds help mitigate the attack, with LM being more effective at lower values of n (nearly as effective as the 'All' strategy).

Effectiveness against the IBT attack in FA is one important aspect of a mitigation measure. Another equally important aspect is FR. We investigate the performance of these mitigation strategies using the same parameter values used in our analysis of FA above, except now h is negative, representing a legitimate prover within the protocol's bound. Figure 8(b) plots FR probabilities for different values of n when $0 > h \geq -0.5$. As n increases, the rejection rate of 'All' increases rapidly all the way to 1.0 while that of SM and LM steadily decreases. Moreover, SM seems to maintain the best FR rates across

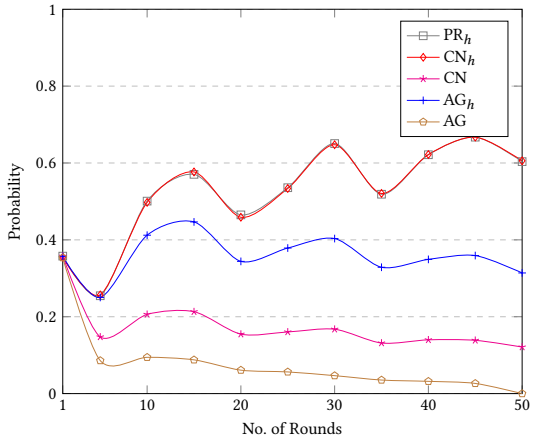


Figure 9: False Acceptance (FA) probability when guessing-ahead with IBT for guessing-ahead (GS1), low noise and large majority (LM) for bit correctness and time tests

all n with some noticeable margin. For $-0.5 > h > -1.0$ (not shown), FR is not an issue as all strategies had zero FR probabilities.

7.2 Guessing-ahead with IBT

As both guessing ahead and IBT-based attacks attempt to manipulate the measured distance computed by the verifier, it would be insightful to see how the two types of attacks interact. Indeed, guessing ahead against a realistic verifier represents a practical setup that system designers and practitioners would want to analyze to assess resilience against such attacks. Therefore, we analyzed the effectiveness of the guessing ahead strategies described above in Section 6.3 against an IBT-vulnerable verifier.

Figure 9 plots FA probability assuming a GS1 prover, low noise and large majority acceptance. By comparing this chart with that of Figure 6a (an abstract verifier), we observe that the IBT vulnerability contributes slightly to the success of guessing ahead attacks by about 5–10% across almost all attacker strategies, except CN_h, which is significantly helped by IBT, to the extent that CN_h becomes as optimal as PR_h. This is significant since it means that, with IBT, the guessing ahead attack can tolerate a level of uncertainty in knowing the exact location of the verifier without sacrificing effectiveness of the attack. For instance, at 10 rounds, CN_h achieves about 50% acceptance with IBT, compared to only 20% without IBT. Gains due to IBT become even higher for larger numbers of rounds.

As for attack detection, AD probability, however, is not affected by exploiting IBT and the results obtained are almost identical to those shown in figure 6b without IBT. This implies that the increased effectiveness of the attacks is indeed due to the attacker being able to manipulate the distance measured by the verifier.

8 CONCLUSION

We presented a model based on Rewriting Logic through which $\mathcal{HK}$ protocols can be faithfully specified and analyzed. The model is characterized by being both expressive and computational, capturing the probabilistic, real-time interactions of these protocols.

Using Maude and PVeStA, probabilistic properties of false acceptance, false rejection and attack detection under different settings and attacker models were mechanically verified through statistical model checking. Verification confirmed very precisely some known results (for which complex manual proofs had to be developed) and enabled the evaluation of new attack strategies in practical settings (noisy channels and resource-constrained verifiers) that are not easily manually analyzable.

There are several possible extensions for future research. First, other scenarios, such as considering probability distributions for the verifier's delays X, Y and Z other than the uniform distribution, can quickly become too complex for manual analysis, but they can still be formally and automatically verified with our model using statistical model checking. Moreover, it would be interesting to model verifiers who are actively trying to counter guessing and IBT attacks, e.g. randomizing challenge timings or observing noticeable variations in measured time-of-flight, and investigate different possible strategies. Another direction for future research is to extend our model and model to allow investigating other kinds of attacks on DB protocols that may involve parties other than the verifier and the prover, such as Mafia and Terrorist fraud attacks. A fourth direction is to develop suitable probabilistic extensions to timed multiset rewriting in which DB protocols (and other probabilistic timed systems) can be specified and probabilistically analyzed.

ACKNOWLEDGMENTS

Alturki is partially supported by KFUPM through his sabbatical project SL161003. Ban Kirigin is supported in part by the Croatian Science Foundation under the project UIP-05-2017-9219. Scedrov is partially supported by ONR. The participation of Kanovich and Scedrov in the preparation of this article was partially within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and partially supported within the framework of a subsidy by the Russian Academic Excellence Project '5-100'. Talcott is partly supported by ONR grant N00014-15-1-2202 and NRL grant N0017317-1-G002.

REFERENCES

[1] Gul Agha. 1986. *Actors: a model of concurrent computation in distributed systems.* MIT Press, Cambridge, MA, USA.

[2] Gul Agha, Carl A. Gunter, Michael Greenwald, Sanjeev Khanna, José Meseguer, Koushik Sen, and Prasanna Thati. 2005. Formal Modeling and Analysis of DoS Using Probabilistic Rewrite Theories. In *International Workshop on Foundations of Computer Security (FCS'05)*. IEEE, Chicago, IL.

[3] Gul Agha, José Meseguer, and Koushik Sen. 2006. PMaude: Rewrite-based Specification Language for Probabilistic Object Systems. *Electronic Notes in Theoretical Computer Science* 153, 2 (2006), 213–239.

[4] Musab A. Alturki and José Meseguer. 2011. PVeStA: A Parallel Statistical Model Checking and Quantitative Analysis Tool. In *Algebra and Coalgebra in Computer Science*, Lecture Notes in Computer Science, Vol. 6859. Springer Berlin / Heidelberg, 386–392.

[5] Musab A. Alturki, José Meseguer, and Carl A. Gunter. 2009. Probabilistic Modeling and Analysis of DoS Protection for the ASV Protocol. *Electron. Notes Theor. Comput. Sci.* 234 (2009), 3–18.

[6] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. 2011. A Framework for Analyzing RFID Distance Bounding Protocols. *J. Comput. Secur.* 19, 2 (April 2011), 289–317. http://dl.acm.org/citation.cfm?id=1971859.1971864

[7] Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gérault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. 2017. A Terrorist-fraud Resistant and Extractor-free Anonymous Distance-bounding Protocol. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17)*. ACM, New York, NY, USA, 800–814.

[8] David Basin, Srdjan Capkun, Patrick Schaller, and Benedikt Schmidt. 2009. Let's Get Physical: Models and Methods for Real-World Security Protocols. In *Theorem Proving in Higher Order Logics: 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1–22.

[9] D. Basin, S. Capkun, P. Schaller, and B. Schmidt. 2011. Formal Reasoning about Physical Properties of Security Protocols. *ACM Transactions on Information and System Security* 14, 2 (2011).

[10] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. 2013. Secure and Lightweight Distance-Bounding. In *Lightweight Cryptography for Security and Privacy*, Springer Berlin Heidelberg, Berlin, Heidelberg, 97–113.

[11] Stefan Brands and David Chaum. 1994. Distance-Bounding Protocols. In *Advances in Cryptology — EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings*, Tor Helleseth (Ed.). Springer, Berlin, Heidelberg, 344–359.

[12] Roberto Bruni and José Meseguer. 2006. Semantic foundations for generalized rewrite theories. *Theor. Comput. Sci.* 360, 1-3 (2006), 386–414.

[13] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. 2007. *All About Maude - A High-Performance Logical Framework*. Lecture Notes in Computer Science, Vol. 4350. Springer-Verlag, Secaucus, NJ, USA.

[14] C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. 2012. Distance Hijacking Attacks on Distance Bounding Protocols. In *2012 IEEE Symposium on Security and Privacy*. 113–127.

[15] Y. G. Dantas, V. Nigam, and I. E. Fonseca. 2014. A Selective Defense for Application Layer DDoS Attacks. In *2014 IEEE Joint Intelligence and Security Informatics Conference*. 75–82.

[16] EasyCrypt. (last accessed: 2018-08-15). https://www.easycrypt.info/trac/.

[17] G. P. Hancke and M. G. Kuhn. 2005. An RFID Distance Bounding Protocol. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. 67–73.

[18] Max Kanovich, Tajana Ban Kirigin, Vivek Nigam, Andre Scedrov, and Carolyn Talcott. 2016. Can we mitigate the attacks on Distance-Bounding Protocols by using challenge-response rounds repeatedly?. In *Workshop on Foundations of Computer Security*.

[19] Max Kanovich, Tajana Ban Kirigin, Vivek Nigam, Andre Scedrov, and Carolyn Talcott. 2017. Time, computational complexity, and probability in the analysis of distance-bounding protocols. *Journal of Computer Security* 25, 6 (2017), 585–630.

[20] Michael Katelman, José Meseguer, and Jennifer Hou. 2008. Redesign of the LMST Wireless Sensor Protocol through Formal Modeling and Statistical Model Checking. In *Proc. of FMOODS '08 (Lecture Notes in Computer Science)*, Vol. 5051. Springer, Berlin, Heidelberg, 150–169.

[21] Chong Hee Kim and Gildas Avoine. 2009. RFID Distance Bounding Protocol with Mixed Challenges to Prevent Relay Attacks. In *Cryptology and Network Security: 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 119–133.

[22] Nirman Kumar, Koushik Sen, José Meseguer, and Gul Agha. 2003. A Rewriting Based Model for Probabilistic Distributed Object Systems.. In *Proc. of FMOODS '03 (Lecture Notes in Computer Science)*, Vol. 2884. Springer, 32–46.

[23] Si Liu, Peter Csaba Ölveczky, Jatin Ganhotra, Indranil Gupta, and José Meseguer. 2017. Exploring Design Alternatives for RAMP Transactions Through Statistical Model Checking. In *Formal Methods and Software Engineering: 19th International Conference on Formal Engineering Methods, ICFEM 2017, Xi'an, China, November 13-17, 2017, Proceedings*, Springer International Publishing, Cham, 298–314.

[24] José Meseguer. 1992. Conditional rewriting logic as a unified model of concurrency. *Theor. Comput. Sci.* 96, 1 (1992), 73–155.

[25] José Meseguer. 1998. Membership algebra as a logical framework for equational specification. In *Proc. WADT'97 (Lecture Notes in Computer Science)*, F. Parisi-Presicce (Ed.), Vol. 1376. Springer, 18–61.

[26] Jorge Munilla and Alberto Peinado. 2008. Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels. *Wireless Communications and Mobile Computing* 8, 9 (2008), 1227–1232.

[27] Dusko Pavlovic and Catherine Meadows. 2010. Bayesian Authentication: Quantifying Security of the Hancke-Kuhn Protocol. *Electronic Notes in Theoretical Computer Science* 265, Supplement C (2010), 97 – 122. Proceedings of the 26th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2010).

[28] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*. 195–212.

[29] P. Schaller, B. Schmidt, D. Basin, and S. Capkun. 2009. Modeling and Verifying Physical Properties of Security Protocols for Wireless Networks. In *2009 22nd IEEE Computer Security Foundations Symposium*. 109–123.

[30] Koushik Sen, Nirman Kumar, Jose Meseguer, and Gul Agha. 2003. *Probabilistic Rewrite Theories: Unifying Models, Logics and Tools.* Technical Report UIUCDCS-R-2003-2347. University of Illinois at Urbana Champaign.