Analysis of Probabilistic Contract Signing

Gethin Norman[1] * and Vitaly Shmatikov[2] **

[1] School of Computer Science, University of Birmingham, Birmingham B15 2TT U.K.
gxn@cs.bham.ac.uk
[2] SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025 U.S.A.
shmat@csl.sri.com

Abstract. We consider the probabilistic contract signing protocol of Ben-Or, Goldreich, Micali, and Rivest as a case study in formal verification of probabilistic security protocols. Using the probabilistic model checker PRISM, we analyse the probabilistic fairness guarantees the protocol is intended to provide. Our study demonstrates the difficulty of combining fairness with timeliness in the context of probabilistic contract signing. If, as required by timeliness, the judge responds to participants' messages immediately upon receiving them, then there exists a strategy for a misbehaving participant that brings the protocol to an unfair state with arbitrarily high probability, unless unusually strong assumptions are made about the quality of the communication channels between the judge and honest participants. We quantify the tradeoffs involved in the attack strategy, and discuss possible modifications of the protocol that ensure both fairness and timeliness.

1 Introduction

Consider several parties on a computer network who wish to exchange some items of value but do not trust each other to behave honestly. *Fair exchange* is the problem of exchanging data in a way that guarantees that either all participants obtain what they want, or none do. *Contract signing* is a particular form of fair exchange, in which the parties exchange commitments to a *contract* (typically, a text string spelling out the terms of the deal). Commitment is often identified with the party's digital signature on the contract. In commercial transactions conducted in a distributed environment such as the Internet, it is sometimes difficult to assess a counterparty's trustworthiness. Contract signing protocols are, therefore, an essential piece of the e-commerce infrastructure.

Contract signing protocols. The main property a contract signing protocol should guarantee is *fairness*. Informally, a protocol between A and B is fair for A if,

* Supported in part by the EPSRC grant GR/N22960.
** Supported in part by DARPA contract N66001-00-C-8015 "Agile Management of Dynamic Collaboration."

in any situation where B has obtained A's commitment, A can obtain B's commitment regardless of B's actions. Ideally, fairness would be guaranteed by the simultaneous execution of commitments by the parties. In a distributed environment, however, simultaneity cannot be assured unless a trusted third party is involved in every communication. Protocols for contract signing are inherently asymmetric, requiring one of the parties to make the first move and thus put itself at a potential disadvantage in cases where the other party misbehaves.

Another important property of contract signing protocols is *timeliness*, or timely termination [ASW00]. Timeliness ensures, roughly, that the protocol does not leave any participant "hanging" in an indeterminate state, not knowing whether the exchange of commitments has been successful. In a timely protocol, each participant can terminate the protocol timely and unilaterally, *e.g.*, by contacting a trusted third party and receiving a response that determines the status of the exchange.

Research on fair contract signing protocols dates to the early work by Even and Yacobi [EY80] who proved that fairness is impossible in a deterministic 2-party contract signing protocol. Since then, there have been proposed randomized contract signing protocols based on a computational definition of fairness [Eve82,EGL85], protocols based on gradual release of commitments [Dam95,BN00], as well as non-probabilistic contract signing protocols that make *optimistic* use of the trusted third party (a.k.a. *judge*). In an optimistic protocol, the trusted third party is invoked only if one of the participants misbehaves [ASW00,GJM99]. In this paper, we focus on *probabilistic* contract signing, exemplified by the probabilistic contract signing protocol of Ben-Or, Goldreich, Micali, and Rivest [BOGMR90] (henceforth, the BGMR protocol).

Analysis technique. Our main contribution is a demonstration of how probabilistic verification techniques can be applied to the analysis of fairness properties of security protocols. While formal analysis of fair exchange is a very active area of research (see the related work section below), formalization and verification of fairness in a probabilistic setting is quite subtle.

We are interested in verifying fairness guarantees provided by the protocol against an arbitrarily misbehaving participant. Therefore, we endow that participant with nondeterministic attacker operations in addition to the probabilistic behaviour prescribed by the protocol specification. The resulting model for the protocol combines nondeterminism and probability, giving rise to a Markov decision process. We discretize the probability space of the model and analyse chosen finite configurations using PRISM, a probabilistic finite-state model checker.

Timeliness and fairness in the BGMR protocol. The original BGMR protocol as specified in [BOGMR90] consists of two phases: the "negotiation" phase of pre-agreed duration, in which participants exchange their partial commitments to the contract, and the "resolution" phase, in which the judge issues decisions in case one or both of the participants contacted him during the negotiation phase. The BGMR protocol does not guarantee timeliness. On the one hand, the negotiation phase should be sufficiently long to enable two honest participants to complete the exchange of commitments without involving the judge. On the other hand, if something goes wrong (*e.g.*, a dishonest party stops responding), the honest party may contact the judge, but then has to wait until the entire period allotted for the negotiation phase is over before he receives the judge's verdict and learns whether the contract is binding on him or not.

We study a variant of the BGMR protocol that attempts to combine fairness with timeliness by having the judge respond immediately to participants' messages, in the manner similar to state-of-the-art non-probabilistic contract signing protocols such as the optimistic protocols of Asokan *et al.* [ASW00] and Garay *et al.* [GJM99]. Our analysis uncovers that, for this variant of the BGMR protocol, fairness is guaranteed only if the judge can establish a communication channel with A, the initiator of the protocol, and deliver his messages faster than A and B are communicating with each other. If the channel from the judge to A provides no timing guarantees, or the misbehaving B controls the network and (passively) delays the judge's messages, or it simply takes a while for the judge to locate A (the judge knows A's identity, but they may have never communicated before), then B can exploit the fact that the judge does not remember his previous verdicts and bring the protocol to an unfair state with arbitrarily high probability.

We quantify the tradeoff between the magnitude of this probability and the expected number of message exchanges between A and B before the protocol reaches a state which is unfair to A. Informally, the longer B is able to delay the judge's messages (and thus continue communicating with A, who is unaware of the judge's attempts to contact him), the higher the probability that B will be able to cheat A.

Related work. A variety of formal methods have been successfully applied to the study of *nondeterministic* contract signing protocols, including finite-state model checking [SM02], alternating transition systems [KR01,KR02], and game-theoretic approaches [BH99,CKS01,BHv02]. None of these techniques, however, are applicable to contract signing in a *probabilistic* setting. Since fairness in protocols like BGMR is a fundamentally probabilistic property, these protocols can only be modelled with a probabilistic formalism such as Markov decision processes and verified only with probabilistic verification tools. Recently, Aldini and Gorrieri [AG02] used a probabilistic process algebra to analyse the fairness guarantees of the probabilistic non-repudiation protocol of Markowitch and Roggeman [MR99] (non-repudiation is a restricted case of contract signing).

Even for non-fairness properties such as secrecy, authentication, anonymity, etc., formal techniques for the analysis of security protocols have focused almost exclusively on nondeterministic attacker models. Attempts to incorporate probability into formal models have been limited to probabilistic characterization of non-interference [Gra92,SG95,VS98], and process formalisms that aim to represent probabilistic properties of cryptographic primitives [LMMS99]. This paper is an attempt to demonstrate how fully automated probabilistic analysis techniques can be used to give a quantitative characterization of probability-based security properties.

2 Probabilistic Model Checking

Probability is widely used in the design and analysis of software and hardware systems: as a means to derive efficient algorithms (*e.g.*, the use of electronic coin flipping in decision making); as a model for unreliable or unpredictable behaviour (*e.g.*, fault-tolerant systems, computer networks); and as a tool to analyse system performance (*e.g.*, the use of steady-state probabilities in the calculation of throughput and mean waiting time). *Probabilistic model checking* refers to a range of techniques for calculating the likelihood of the occurrence of certain events during the execution of such system, and can be useful to establish performance measures such as "shutdown occurs with probability at most 0.01" and "the video frame will be delivered within 5ms with probability at least 0.97". The system is usually specified as state transition system, with probability measures on the rate of transitions, and a probabilistic model checker applies algorithmic techniques to analyse the state space and calculate performance measures.

In the distributed scenario, in which concurrently active processors handle a great deal of unspecified nondeterministic behaviour exhibited by their environment, the state transition systems must include both probabilistic and nondeterministic behaviour. A standard model of such systems are Markov decision processes (MDPs) [Der70]. Properties of MDPs can be specified in the probabilistic branching-time temporal logic PCTL [HJ94,BdA95] which allows one to express properties such as "under any scheduling of nondeterministic choices, the probability of ϕ holding until ψ is true is *at least 0.78/at most 0.04*".

2.1 PRISM model checker

We use PRISM [KNP02,Pri], a probabilistic model checker developed at the University of Birmingham. The current implementation of PRISM supports the analysis of *finite*-state probabilistic models of the following three types: discrete-time Markov chains, continuous-time Markov chains and Markov decision processes. These models are described in a high-level language, a variant of reactive modules [AH99] based on guarded commands. The basic components of the language are *modules* and *variables*. A system is constructed as a number of modules which can interact with each other. A module contains a number of variables which express the state of the module, and its behaviour is given by a set of guarded commands of the form:

$$[]\ \texttt{<guard>} \rightarrow \texttt{<command>};$$

The guard is a predicate over the variables of the system and the command describes a transition which the module can make if the guard is true (using primed variables to denote the next values of variables). If a transition is probabilistic, then the command is specified as:

$$\texttt{<prob> : <command>} + \cdots + \texttt{<prob> : <command>}$$

PRISM accepts specifications in either PCTL, or CSL logic depending on the model type. This allows us to express various probabilistic properties such as "some event happens with probability 1", and "the probability of cost exceeding C is 95%". The model checker then analyses the model and checks if the property holds in each state. In the case of MDPs, specifications are written in the logic PCTL, and for the analysis PRISM implements the algorithms of [HJ94,BdA95,BK98].

3 BGMR Protocol

The objective of the probabilistic contract signing protocol of Ben-Or, Goldreich, Micali, and Rivest [BOGMR90] (the BGMR protocol) is to enable two parties, A and B, to exchange their commitments to a pre-defined contract C. It is assumed that there exists a third party, called the *judge*, who is trusted by both A and B. The protocol is *optimistic*, *i.e.*, an honest participant following the protocol specification only has to invoke the judge if something goes wrong, *e.g.*, if the other party stops before the exchange of commitments is complete (a similar property is called *viability* in [BOGMR90]). Optimism is a popular feature of fair exchange protocols [Mic97,ASW97,ASW00]. In cases where both signers are honest, it enables contract signing to proceed without participation of a third party, and thus avoids communication bottlenecks inherent in protocols that involve a trusted authority in every instance.

3.1 Privilege and fairness

In the BGMR protocol, it can never be the case that the contract is binding on one party, but not the other. Whenever the judge declares a contract binding, the verdict always applies to *both* parties. For brevity, we will refer to the judge's ruling on the contract as *resolving* the contract.

Privilege is a fundamental notion in the BGMR protocol. A party is privileged if it has the power to cause the judge to rule that the contract is binding. The protocol is unfair if it reaches a state where one party is privileged (*i.e.*, it can cause the judge to declare the contract binding), and the other is not.

Definition 1 (Probabilistic fairness).
A contract signing protocol is $(v, \epsilon)-$fair *for* A *if, for any contract* C*, if* A *follows the protocol, then at any step of the protocol in which the probability that* B *is privileged is greater than* v*, the conditional probability that* A *is* not *privileged given that* B *is privileged is at most* ϵ*.*

The fairness condition for B is symmetric.

Probabilistic fairness implies that at any step of the protocol where one of the parties has acquired the evidence that will cause the judge to declare the contract binding with probability x, the other party should possess the evidence that will cause the judge to issue the same ruling with probability of no less than $x - \epsilon$. Informally, ϵ can be interpreted as the maximum *fairness gap* between A and B permitted at any step of the protocol.

3.2 Main protocol

Prior to initiating the protocol, A chooses a probability v which is sufficiently small so that A is willing to accept a chance of v that B is privileged while A is not.

A also chooses a value $\alpha > 1$ which quantifies the "fairness gap" (see section 3.1) as follows: at each step of the protocol the conditional probability that A is privileged given that B is privileged should be at least $\frac{1}{\alpha}$, unless the probability that B is privileged is under v. B also chooses a value $\beta > 1$ such that at any step where A is privileged, the conditional probability that B is privileged should be at least $\frac{1}{\beta}$. Both parties maintain counters, λ_a and λ_b, initialized to 0.

A's commitment to C has the form "$\mathrm{sig}_A($*With probability* 1*, the contract* C *shall be valid*$)$". B's commitment is symmetric. It is assumed that the protocol employs an unforgeable digital signature scheme.

All messages sent by A in the main protocol have the form "$\mathrm{sig}_A($*With probability* p*, the contract* C *shall be valid*$)$". Messages sent by B have the same form and are signed by B. If both A and B behave correctly, at the end of the main protocol A obtains B's commitment to C, and vice versa. At the abstract level, the main flow of the BGMR protocol is as follows:

$$
\begin{array}{ll}
A \rightarrow B & \mathrm{sig}_A(\textit{With prob. } p_1^a\textit{, the contract } C \textit{ shall be valid}) = m_1^a \\
A \leftarrow B & \mathrm{sig}_B(\textit{With prob. } p_1^b\textit{, the contract } C \textit{ shall be valid}) = m_1^b \\
 & \vdots \\
A \rightarrow B & \mathrm{sig}_A(\textit{With prob. } p_i^a\textit{, the contract } C \textit{ shall be valid}) = m_i^a \\
 & \vdots \\
A \rightarrow B & \mathrm{sig}_A(\textit{With prob. } 1\textit{, the contract } C \textit{ shall be valid}) = m_n^a \\
A \leftarrow B & \mathrm{sig}_B(\textit{With prob. } 1\textit{, the contract } C \textit{ shall be valid}) = m_n^b
\end{array}
$$

In its first message m_1^a, A sets $p_1^a = v$.

Consider the ith round of the protocol. After receiving message $\mathrm{sig}_A($*With prob.* p_i^a*, the contract* C *shall be valid*$)$ from A, honest B checks whether $p_i^a \geq \lambda_b$. If not, B considers A to have stopped early, and contacts the judge for resolution of the contract as described in section 3.3. If the condition holds, B computes $\lambda_b = \min(1, p_i^a \cdot \beta)$, sets $p_i^b = \lambda_b$ and sends message $\mathrm{sig}_B($*With prob.* p_i^b*, the contract* C *shall be valid*$)$ to A.

The specification for A is similar. Upon receiving B's message with probability p_i^b in it, A checks whether $p_i^b \geq \lambda_a$. If not, A contacts the judge, otherwise he updates $\lambda_a = \max(v, \min(1, p_i^b \cdot \alpha))$, sets $p_{i+1}^a = \lambda_a$ and sends message $\mathrm{sig}_A($*With prob.* p_{i+1}^a*, the contract* C *shall be valid*$)$, initiating a new round of the protocol.

The main protocol is optimistic. If followed by both participants, it terminates with both parties committed to the contract.

3.3 The judge

Specification of the BGMR protocol assumes that the contract C defines a cutoff date D. When the judge is invoked, he does nothing until D has passed, then examines the message of the form sig_X(*With prob. p, the contract C shall be valid*) supplied by the party that invoked it and checks the validity of the signature.

If the judge has not resolved contract C before, he flips a coin, *i.e.*, chooses a random value ρ_C from a uniform distribution over the interval $[0,1]$. If the contract has been resolved already, the judge retrieves the previously computed value of ρ_C. In either case, the judge declares the contract binding if $p \geq \rho_C$ and cancelled if $p < \rho_C$, and sends his verdict to the participants. To make the protocol more efficient, ρ_C can be computed as $f_r(C)$, where r is the judge's secret input, selected once and for all, and f_r is the corresponding member of a family of pseudo-random functions [GGM86]. This enables the judge to produce the same value of ρ_C each time contract C is submitted without the need to remember his past flips. The judge's procedure can thus be implemented in constant memory.

Observe that even though the judge produces the same value of ρ_C each time C is submitted, the judge's *verdict* depends also on the value of p in the message submitted by the invoking party and, therefore, may be different each time. If the judge is optimized using a pseudo-random function with a secret input to work in constant memory as described above, it is impossible to guarantee that he will produce the same verdict each time. To do so, the judge needs to remember the first verdict for each contract ever submitted to him, and, unlike ρ_C, the value of this verdict cannot be reconstructed from subsequent messages related to the same contract.

3.4 Timely BGMR

Asokan *et al.* [ASW00] define *timeliness* as "one player cannot force the other to wait for any length of time—a fair and timely termination can be forced by contacting the third party." The BGMR protocol as specified in sections 3.2 and 3.3 does not guarantee timeliness in this sense. To accommodate delays and communication failures on a public network such as the Internet, the duration of the negotiation phase D should be long. Otherwise, many exchanges between honest parties will not terminate in time and will require involvement of the judge, making the judge a communication bottleneck and providing no improvement over a protocol that simply channels all communication through a trusted central server.

If D is long, then an honest participant in the BGMR protocol can be left "hanging" for a long time. Suppose the other party in the protocol stops communicating. The honest participant may contact the judge, of course, but since the judge in the original BGMR protocol does not flip his coin until D has passed, this means that the honest party must wait the *entire* period allotted for negotiation before he learns whether the contract will be binding on him or not. This lack of timeliness is inconvenient and potentially problematic if the contract requires resource commitment from the participants or relies on time-sensitive data.

In this paper, we investigate a variant of BGMR that we call TBGMR (for "Timely BGMR"). The only difference between BGMR and TBGMR is that, in

TBGMR, the judge makes his decision *immediately* when invoked by one of the protocol participants. Once the verdict is announced and reaches an honest participant, the latter stops communicating. The rest of the protocol is as specified in sections 3.2 and 3.3. TBGMR protocol is timely. Any party can terminate it unilaterally and obtain a binding verdict at any point in the protocol without having to wait for a long time.

4 Model

We formalize both the BGMR and TBGMR protocols as Markov decision processes. We then use PRISM on a discretized model to determine if TBGMR is fair.

4.1 Overview of the model

First, we model the normal behaviour of protocol participants and the judge according to the BGMR protocol specification, except that the judge makes his coin flip and responds with a verdict immediately. A dishonest participant might be willing to deviate from the protocol in an attempt to cheat the honest participant. We assume that B is the dishonest participant and A the honest one. We equip B with an additional set of dishonest actions, any of which he can take nondeterministically at any point in the protocol. To obtain a finite probabilistic model, we fix the parameters chosen by the participants (see section 3.2), and discretize the probability space of the judge's coin flips.

Modelling the dishonest participant. Conventional formal analysis of security protocols is mainly concerned with security against the so called *Dolev-Yao attacker*, following [DY83]. A Dolev-Yao attacker is a nondeterministic process that has complete control over the communication network and can perform any combination of a given set of attacker operations, such as intercepting any message, splitting messages into parts, decrypting if he knows the correct decryption key, assembling fragments of messages into new messages and replaying them out of context, etc.

We assume that the digital signature scheme employed by the protocol to authenticate messages between participants is secure. Therefore, it is impossible for the misbehaving participant to forge messages from the honest participant or modify their content. We will also assume that the channels between the participants and the judge are *resilient*: it is possible for the attacker to delay messages and schedule them out of order, but he must eventually deliver every message to its intended recipient.

Dishonest actions available to a misbehaving participant are thus limited to i) invoking the judge even though the honest party has not stopped communicating in the main flow of the protocol, and ii) delaying and re-ordering messages between the judge and the honest party. The misbehaving participant can nondeterministically attempt any of these actions at any point in the protocol. When combined with the probabilistic behaviour of the judge, the nondeterminism of the dishonest participant gives rise to a Markov decision process.

Modelling fairness. To model fairness, we compute the maximum probability of reaching a state where the dishonest participant is privileged and the honest one is not. Note that this probability is *not* conditional on the judge's coin not having been flipped yet, because the dishonest participant may choose to contact the judge even after the coin has been flipped, ρ_C computed and verdict rendered. In contrast, the proof of fairness in [BOGMR90] calculates this probability under the assumption that the coin has not been flipped yet.

4.2 Analysis technique

We now describe our method for modelling the protocol in PRISM's input description language. Since PRISM is currently only applicable to finite configurations and the input language allows only integer valued variables, to model the protocol there are two simplifications we need to make.

First, we must discretize the judge's coin by fixing some $N \in \mathbb{N}$ and supposing that when the judge flips the coin, it takes the value i/N for $i = 1, \dots, N$ with probability $1/N$. Second, we must fix the parameters of the parties, namely v, α and β.

Once the parameters v, α and β are fixed, the possible messages that can be sent between the parties and the ordering of the messages are predetermined. More precisely, as shown in fig. 1, the probability values included in the messages between the parties are known. Note that, for $v, \alpha, \beta > 0$, these values converge to 1 and there are only finitely many distinct values. Therefore, with a simple script, we can calculate all the probability values included in the messages of the two parties. Now, to model the parties in PRISM, we encode the state of each party as the probability value included in the last message the party sent to the other party, *i.e.*, the states of A and B are identified by the current values of λ_a and λ_b respectively. Formally, the states of each party range from 0 up to k, for some k such that $p_i^a, p_i^b = 1$ for all $i \geq k$, where A is in state 0 when A has sent no messages to B, and in state $i \in \{1, \dots, k\}$ when the last message A sent to B included the probability value $\min(1, v \cdot \alpha^{i-1} \cdot \beta^{i-1})$. Similarly, B is in state 0 when B has not sent any messages to A, and in state $i \in \{1, \dots, k\}$ when the last message B sent to A included the probability value $\min(1, v \cdot \alpha^{i-1} \cdot \beta^{i})$.

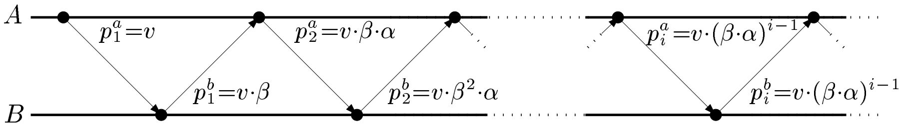

Fig. 1. Probability values included in the messages sent between the parties.

Following this construction process, in the case when $N = 100$, the specification of the BGMR protocol used as input into PRISM is given in fig. 2. Note

```
module

  lambdaA : [0..k]; // probability value in the last message A sent to B
  lambdaB : [0..k]; // probability value in the last message B sent to A
  turn : [0..1]; // which party sends message next (0 – party A and 1 – party B)
  c : [0..1];  // status of the coin (0 – not flipped and 1 – flipped)
  rho : [0..100]; // value of the coin (rho=x and c=1 means its value is x/100)

  // parties alternately send messages and stop when the coin has been flipped
  [] turn=0 ∧ c=0 → lambdaA'=min(k, lambdaA + 1) ∧ turn'=1;
  [] turn=1 ∧ c=0 → lambdaB'=min(k, lambdaB + 1) ∧ turn'=0;
  // flip coin any time after a message has been sent
  [] c=0 ∧ (lambdaA>0 ∨ lambdaB>0) → 1/100 : (c'=1) ∧ (rho'=1)
                                      +1/100 : (c'=1) ∧ (rho'=2)
                                                   ⋮
                                      +1/100 : (c'=1) ∧ (rho'=100);

endmodule
```

Fig. 2. Prism code for BGMR protocol.

that there is a nondeterministic choice as to when the judge's coin is flipped, *i.e.*, when a party first sends a message to the judge asking for a verdict. Furthermore, once the coin is flipped, the parties cannot send any messages to each other, that is, this model corresponds to the original protocol.

In the timely variant of the protocol (TBGMR), if the misbehaving participant is capable of delaying messages from the judge, then the parties may continue sending messages to each other even after the coin has been flipped. To model this, the two lines corresponding to the parties sending messages are replaced with:

```
// parties can continue sending messages after the coin has been flipped
[] turn=0 → lambdaA'=min(k, lambdaA + 1) ∧ turn'=1;
[] turn=1 → lambdaB'=min(k, lambdaB + 1) ∧ turn'=0;
```

Recall, if the coin has been flipped and the last messages sent by parties A and B include the probability values p^a and p^b respectively, then in the current state of the protocol B is privileged and A is not if and only if the value of the coin is in the interval $(p_b, p_a]$. Therefore, in the PRISM model we can specify the set of states where B is privileged and A is not. Let's call this set of states **unfair**. We then use PRISM to calculate the *maximum probability*, from the initial state (where no messages have been sent and the coin has not been flipped), of reaching a state in **unfair**. Table 1 gives the summary of the results obtained using PRISM when considering a number of different parameters for both the BGMR and TBGMR versions of the protocol.

v	α	β	N	number of states	**maximum probability of reaching a state where only B is privileged**	
					BGMR	**TBGMR**
0.1	1.1	1.05	10	408	**0.1000**	**0.8000**
			100	3,738	**0.1000**	**0.7000**
			1,000	37,038	**0.1000**	**0.7080**
0.1	1.1	1.01	10	518	**0.1000**	**0.9000**
			100	4,748	**0.1000**	**0.9000**
			1,000	47,048	**0.1000**	**0.9180**
0.01	1.01	1.005	10	6,832	**0.1000**	**0.6000**
			100	62,722	**0.0100**	**0.6600**
			1,000	621,622	**0.0100**	**0.6580**
0.01	1.01	1.001	10	9,296	**0.1000**	**1.0000**
			100	85,346	**0.0100**	**0.9400**
			1,000	845,846	**0.0100**	**0.9070**
0.001	1.001	1.0005	10	101,410	**0.1000**	**0.6000**
			100	931,120	**0.0100**	**0.7200**
			1,000	9,228,220	**0.0010**	**0.6840**
0.001	1.001	1.0001	10	138,260	**0.1000**	**0.9000**
			100	1,269,470	**0.0100**	**0.8700**
			1,000	12,581,570	**0.0010**	**0.9010**

Table 1. Model checking results obtained with PRISM

5 Analysis Results

The probability of reaching a state in **unfair** is maximized, over all nondeterministic choices that can be made by a misbehaving B, by the following strategy. As soon as B receives the first message from A, he invokes the judge, pretending that A has stopped communicating and presenting A's message to the judge. In TBGMR, this will cause the judge to flip the coin and announce a verdict immediately. B delays the judge's announcement message to A, and keeps exchanging messages with A in the main flow of the protocol. After each message from A, B invokes the judge and presents A's latest message, until one of them results in a positive verdict. Depending on his choice of β, B can steer the protocol to a state unfair to A with arbitrarily high probability.

Table 1 lists the maximum probabilities that can be achieved by this strategy in the TBGMR protocol. Recall that the results obtained with PRISM are for a simplified model, where the coin is discretized. Nevertheless, due to the simplicity of this strategy, we were able to write a simple MATLAB script which calculated the probability of reaching a state which is unfair to A under this strategy for a general coin, that is, a coin whose flips take a value uniformly chosen from the $[0, 1]$ interval.

Fig. 3 (left chart) shows the probability, for various choices of parameters, of reaching a state in which B is privileged and A is not. Note that B can bring this probability arbitrarily close to 1 by choosing a small β. We assume that the value of β is constant and chosen by B beforehand. In practice, B can decrease his β adaptively. A may respond in kind, of course, but then the protocol will crawl to a halt even if B is not cheating.

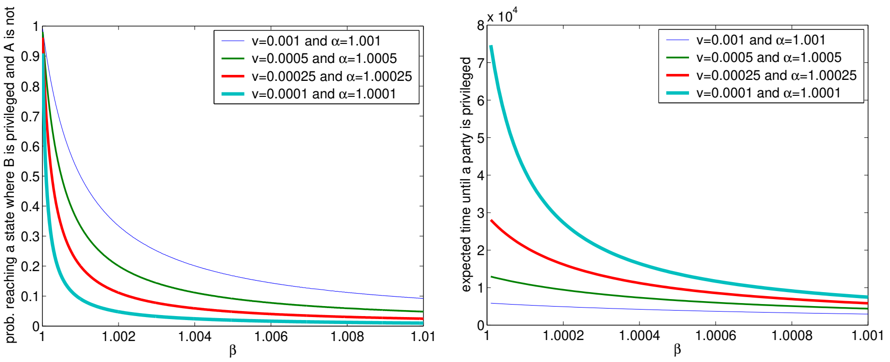

Fig. 3. Probability and Expected time of B's win, depending on β

This attack on TBGMR is feasible because the judge remembers (or reconstructs, using a pseudo-random function with a secret input—see section 3.3) only the value of his coin flip, ρ_C, and *not* his previous verdicts. Therefore, B induces the coin flip as soon as possible, and then gets the judge to produce verdict after verdict until the probability value contained in A's message is high enough to result in a positive verdict.

Accountability of the judge. Under the assumption that the channels are resilient (*i.e.*, eventual delivery is guaranteed), at some point A will receive all of the judge's verdicts that have been delayed by B. Since these verdicts are contradictory (some declare the contract binding, and some do not), A may be able to argue that somebody—either B, or the judge, or both—misbehaved in the protocol. Note, however, that the protocol specification does *not* require the judge to produce consistent verdicts, only consistent values of ρ_C.

To repair this, the protocol may be augmented with "rules of evidence" specifying what constitutes a violation (see also the discussion in section 6). Such rules are not mentioned in [BOGMR90] and appear to be non-trivial. A discussion of what they might look like is beyond the scope of this paper.

5.1 Attacker's tradeoff

Although a Dolev-Yao attacker is assumed to be capable of delaying messages on public communication channels, in practice this might be difficult, especially if long delays are needed to achieve the attacker's goal. Therefore, it is important to quantify the relationship between B's probability of winning (bringing the protocol into a state that's unfair to A), and how long B needs to delay the judge's messages to A. As a rough measure of time, we take the expected number of message exchanges between A and B. The greater the number of messages A has to send before B reaches a privileged state, the longer the delay.

We wrote a MATLAB script to calculate the expected number of messages sent before a party becomes privileged. The results are shown in fig. 3 (right chart) for various values of v and α as β varies. As expected, the lower β, the greater the number of messages that the parties have to exchange before B becomes privileged. Recall, however, that lower values of β result in higher probability that B *eventually* wins (left chart on fig. 3).

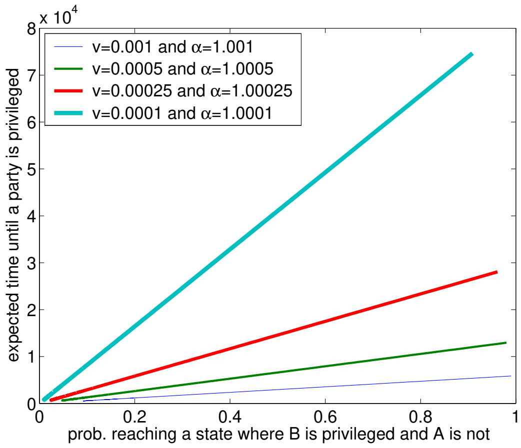

Fig. 4. Expected time to B's win vs. probability of B's win

The misbehaving participant thus faces a tradeoff. The higher the probability of winning, the longer the judge's messages to the honest participant must be delayed. This tradeoff is quantified by the chart in fig. 4. As the chart demonstrates, there is a linear tradeoff for a misbehaving B between the expected time to win and the probability of winning. If B is not confident in his ability to delay the judge's messages to A and/or exchange messages with A fast enough (before the judge's verdict reaches A), B can choose a large value of β and settle for a shorter delay and thus lower probability of steering the protocol into an unfair state.

6 Making BGMR Protocol Timely

As demonstrated in section 5, the BGMR protocol cannot be made timely by simply having the judge flip his coin immediately after he is invoked, because then fairness is lost. In this section, we discuss modifications to the protocol that may enable it to provide both timeliness and fairness.

Fast and secure communication channels. The attack described in section 5 can be prevented if the protocol requires the communication channel between the judge and the honest participant to be of very high quality. More precisely, both fairness and timeliness will be guaranteed if B is prevented from delaying the judge's messages to A. Also, the judge's channel to A must be faster than the channel between A and B. This ensures that A will be notified of the judge's verdict immediately after the judge flips his coin and will stop communicating with B.

While a high-quality channel *from* A to the judge might be feasible, it is significantly more difficult to ensure that the channel from the judge *to* A is faster than the channel between A and B. The judge is necessarily part of the infrastructure in which the protocol operates, servicing multiple instances of the protocol at the same time. Therefore, it is reasonable to presume that the judge is always available and responsive. On the other hand, A is simply one of many participants using the protocol, may not be easy to locate on a short notice, may not be expecting a message from the judge before the cutoff date D, etc. On a public network, it is usually much easier for a user to contact the authorities than for the authorities to locate and contact the user.

Judge with unbounded memory. The attack also succeeds because the judge does not remember his past verdicts, only the value of his coin flip ρ_C. If the judge is made to remember his first verdict, and simply repeat it in response to all subsequent invocations regarding the same contract, the attack will be prevented. Note, however, that such a judge will require unbounded memory, because, unlike ρ_C, the value of the first verdict cannot be reconstructed from subsequent evidence using a pseudo-random function. Therefore, the judge will need to store the verdict he made for every instance of the protocol he has ever been asked to resolve in case he is invoked again regarding the same instance.

A possible optimization of this approach is to expunge all verdicts regarding a particular contract from the judge's memory once the cutoff date for that contract has passed. This introduces additional complications, however, since participants are permitted to ask the judge for a verdict even after the cutoff date. If the verdict in this case is constructed from ρ_C and the newly presented evidence, it may be inconsistent with the verdicts the judge previously announced regarding the same contract. To avoid this, the old verdicts must be stored forever and the judge's memory must be infinite, regardless of the quality of the communication channels between the parties and the judge.

Signed messages to the judge. If all invocations of the judge are signed by the requesting party, and that signature is included in the judge's verdict, then A will be able to prove B's misbehaviour (that B invoked the judge more than once) when all of the judge's verdicts finally reach him after having been delayed by B. If invoking the judge more than once is construed as a violation, however, the protocol might be problematic for an honest B in case there is a genuine delay on the channel between A and B. Such a delay may cause an honest B to time out, invoke the judge, and then, after A's message with a new probability value finally arrives, invoke the judge again, which would be interpreted as an attempt to cheat.

Combining fairness with timeliness seems inherently difficult in the case of probabilistic contract signing. The approaches outlined above demand either extremely stringent requirements on the communication channels, which would limit applicability of the protocol, or introduction of the additional "rules of evidence" that define what is and what is not a violation. Such rules are not included in the protocol specification [BOGMR90] and would have to be designed from scratch. Designing an appropriate set of rules appears difficult, as they should accommodate legitimate scenarios (a participant should be permitted to request a verdict from the judge more than once, because the judge's messages might have been lost or corrupted in transmission), while preventing a malicious party from repeatedly contacting the judge until the desired verdict is obtained.

7 Conclusions

We presented a case study, demonstrating how formal analysis techniques can be used to verify properties of probabilistic security protocols. We analysed a timely variant of the probabilistic contract signing of Ben-Or, Goldreich, Micali, and Rivest using PRISM, a probabilistic model checker, and showed that there exists an attack strategy for a misbehaving participant that brings the protocol into an unfair state with arbitrarily high probability. We also quantified the attacker's tradeoff between the probability of winning and the need to delay messages from the judge to the honest participant. Designing a probabilistic contract signing protocol that is both timely and fair is a challenging task. Our case study, in addition to demonstrating feasibility of probabilistic model checking as an analysis technique for security protocols, is a step in this direction.

References

[AG02] A. Aldini and R. Gorrieri. Security analysis of a probabilistic non-repudiation protocol. In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 17–36. Springer-Verlag, 2002.

[AH99] R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15:7–48, 1999.

[ASW97] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 7–17, 1997.

[ASW00] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Selected Areas in Communications*, 18(4):593–610, 2000.

[BdA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *LNCS*, pages 499–513. Springer-Verlag, 1995.

[BH99] L. Buttyán and J.-P. Hubaux. Toward a formal model of fair exchange — a game theoretic approach. Technical Report SSC/1999/39, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 1999.

[BHv02] L. Buttyán, J.-P. Hubaux, and S. Čapkun. A formal analysis of Syverson's rational exchange protocol. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 193–205, 2002.

[BK98] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.

[BN00] D. Boneh and M. Naor. Timed commitments. In *Proc. CRYPTO'00*, volume 1880 of *LNCS*, pages 236–254. Springer-Verlag, 2000.

[BOGMR90] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.

[CKS01] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 176–185, 2001.

[Dam95] I.B. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. Cryptology*, 8(4):201–222, 1995.

[Der70] C. Derman. *Finite-State Markovian Decision Processes.* New York: Academic Press, 1970.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[Eve82] S. Even. A protocol for signing contracts. Technical Report 231, Computer Science Dept., Technion, Israel, 1982.

[EY80] S. Even and Y. Yacobi. Relations among public key signature schemes. Technical Report 175, Computer Science Dept., Technion, Israel, 1980.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GJM99] J.A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Proc. CRYPTO'99*, volume 1666 of *LNCS*, pages 449–466. Springer-Verlag, 1999.

[Gra92] J.W. Gray. Toward a mathematical foundation for information flow security. *J. Computer Security*, 1(3):255–294, 1992.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[KNP02] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS'02*, volume 2324 of *LNCS*, pages 200–204. Springer-Verlag, 2002.

[KR01] S. Kremer and J.-F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *Proc. CONCUR'01*, volume 2154 of *LNCS*, pages 551–565. Springer-Verlag, 2001.

[KR02] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 206–220, 2002.

[LMMS99] P. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security analysis. In *Proc. World Congress on Formal Methods*, volume 1708 of *LNCS*, pages 776–793. Springer-Verlag, 1999.

[Mic97] S. Micali. Certified e-mail with invisible post offices. Presented at RSA Security Conference, 1997.

[MR99] O. Markowitch and Y. Roggeman. Probabilistic non-repudiation without trusted third party. In *Proc. 2nd Conference on Security in Communication Networks*, 1999.

[Pri] PRISM web page. http://www.cs.bham.ac.uk/~dxp/prism/.

[SG95] P. Syverson and J.W. Gray. The epistemic representation of information flow security in probabilistic systems. In *Proc. 8th IEEE Computer Security Foundations Workshop*, pages 152–166, 1995.

[SM02] V. Shmatikov and J.C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 283(2):419–450, 2002.

[VS98] D. Volpano and G. Smith. Probabilistic non-interference in a concurrent language. In *Proc. 11th IEEE Computer Security Foundations Workshop*, pages 34–43, 1998.