Cyberdice: peer-to-peer gambling in the presence of cheaters

Frank Stajano and Richard Clayton

University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge, CB3 0FD, United Kingdom

Abstract. We describe a simple gambling game in which n participants each put down a fixed amount of money and one of them, selected at random, wins and takes it all. We describe how this game can be operated in cyberspace, without knowing anything about the other participants except for the bit strings they transmit. We show how the genuine winner can convert the bit strings back into money, without any other gambler or eavesdropper being able to do so before her. We also show that it is possible to have confidence in the fair running of the game even if all the other participants, including the dealer, are crooked and are prepared to manipulate the protocol to their advantage. The paper initially develops a naïve protocol for running the game, and shows various ways in which a gambler can cheat by ceasing to send messages once it is clear that she is losing. We incrementally build this up into a protocol that resists drop-outs, collusion and dishonesty from all players, by relying on the honest behaviour of some non-gambling 'issuers' whose role is to convert currency into bit strings and vice versa.

1 Introduction

Five people sit around a table and throw dice in turn: whoever gets the highest score is the winner.[1] This is a simple game of chance that designates a winner out of a group of people.

Now consider bringing this game to cyberspace. Participants can't see each other throwing the dice: all they know about each other is that they send and receive bit strings. They may be strongly motivated to cheat: imagine they are playing for money, with each gambler paying a fixed fee to play, and winner take all. You never see their faces: they could be spam lords, phishing operators or other remorseless online criminals, so assume that they will definitely cheat if given the chance—simply asking participants to publish the outputs of their individual random number generators won't do.

The problem to be solved is twofold: firstly, we seek a protocol that allows a group of adversarial players to determine a winner fairly, even in the presence of cheaters and even if they know nothing about each other except the bit strings

[1] Yes, there might be *ex-aequo* winners. For the moment, imagine dice with 2^{32} sides so that this occurrence is unlikely. We shall properly fix this problem later.

[Authors' final version as submitted to Springer] (rev. 93)
In B. Christianson et al. (Eds.), *Proc. Security Protocols Workshop 2008*, © Springer LNCS.

they exchange. Secondly, we need a protocol and a digital payment system that will ensure that the winner gets paid by the losers, even if the losers are dishonest and have no intention of honouring their debts.

This paper offers three main contributions to security research. The first is Cyberdice, a peer-to-peer gambling protocol that satisfies the above requirements for a game. The second is a sub-protocol that addresses the digital payment issue: how to 'put money on the table' in the form of a bit string while ensuring that only the winner (unknown at that point) will be able to redeem it. The third contribution, at a more general level, is the discussion of a class of protocol attacks in which malicious principals choose at each step whether to continue following the prescribed protocol or not.

The paper starts by setting out our requirements for the protocol (Section 2) and by introducing our approach (Section 3). In Section 4 we present 'Cyberdice 0.1', a naïve version of the protocol that fails to prevent cheating if players choose not to follow the protocol to its conclusion. In Section 5 we change our requirements to address this type of cheating and discuss the true difficulty of the problem we have posed. In Section 6 we present 'Cyberdice 0.9', a modified version of Cyberdice 0.1 resistant to players dropping out. However this protocol can still be simplified and enhanced, so in Section 7 we present the full detail of 'Cyberdice 1.0', which meets all our extended requirements and, as part of it, we present the sub-protocol for ensuring that the winner gets paid (Section 7.4). We finish by discussing related work in Section 8 and offer some concluding thoughts in Section 9. For convenience, an appendix offers a one-page reference to the sequence of messages exchanged during the protocol.

2 Why is peer-to-peer online gambling hard?

Gambling in cyberspace is hard, because it very difficult to know who you are playing with, or whether you can locate them in the future to hold them to account for their past actions. No player will be prepared to trust any other player sufficiently that they would let them 'run' the game; for the same reason, the protocol cannot rely on a trusted third party to select the winner, because that trust might be misplaced.

If any third parties such as banks are involved in supplying digital currency, we do not want to rely on such third parties to act as trusted casino operators—firstly because we assumed that the players would not trust anyone to roll dice fairly; secondly because we do not want those pseudo-banks to be tainted by the stigma of the gambling act. Their only responsibility must be to issue bit strings that can be used as currency and then exchange them back into currency according to a previously agreed deterministic criterion for accepting or rejecting the claim. It must be possible for the currency-issuing third party to convince an external auditor that they acted appropriately and in strict agreement with that stated deterministic criterion.

The protocol must be fair, in the sense of selecting a winner with equal probability from amongst all the participants, regardless of whether any or all of

them misbehave. It must still be possible for any player to win even if all of the other players collude, and for that matter to win with exactly the same odds as if everyone played honestly.

It must be possible to play over an insecure communication medium such as, say, Usenet News: we assume that any message may be overheard by any player (or third party) and that players or third parties may act as Dolev-Yao adversaries capable of intercepting, deleting and rewriting any messages.

However, to have some chance of constructing a solution, we assume that the nodes are capable of running strong cryptographic algorithms that the adversaries cannot break, and that the internal actions and secrets of the nodes cannot be observed.

Finally, it must be possible for anyone, including third parties as well as the participants themselves, to reach the same conclusion about who won the game by observing all the messages sent and received by the players.

Cyberdice honours all of the above constraints.

3 Cyberdice: the core ideas

3.1 Handling money in cyberspace

In order to take part in the game, gamblers first need to put their money on the table. Doing so in cyberspace is not trivial: if a bit string can be converted into currency by any beneficiary, then, as soon as a player publishes it, any other dishonest player (or eavesdropper) can grab the digital cash string and spend it. On the other hand it is not possible for the player to 'put on the table' a digital cheque written out to a specific beneficiary, because at that stage the winner of the game is yet to be decided.

In Cyberdice, the idea is to pay a trusted third party (acting more as an escrow agent than as a bank) to issue a bit string that can only be redeemed for currency (minus a small handling fee) by the winner of a designated game. Such third party is called an *issuer*. Anyone who overhears the string can bring it to the originating issuer agent for payment; but the issuer will only pay after being presented with undisputable evidence that the bearer won the game mentioned in the string itself.

There are therefore three kinds of participants: issuers, dealer and gamblers (the latter two also referred to collectively as 'players'), whose roles and functions are described in greater detail below.

3.2 The issuers

The Cyberdice protocol does not require an issuer to overlook or run the game in any way: in fact it even allows each gambler to use a different issuer. The issuers merely act as 'transducers' between the world of currency and that of bit strings: their only job is to accept money in escrow, publish signed strings and redeem bit strings with certain properties against deposited money, after

verifying the credentials of the redeemer. The winner collects their takings from the relevant issuer(s) in turn, by presenting the necessary evidence. By staying outside the game itself, the issuers cannot lose money—indeed we expect them to charge a fee for each conversion operation they perform. In essence, the main requirement is that the chosen issuers be 'well known' and trusted by their customers to honour their contracts. We also assume that issuers are robust against Denial Of Service (DOS) attacks and therefore that they will always be contactable within a reasonable time (which we will not presume is true for the players, whose cyber-existence is ephemeral).

3.3 The players

Unlike the issuers, the players—both the dealer and the gamblers—are not trusted by anyone. Hence they are required to put their money into escrow before the real action starts. Dealers[2] collect a fee from anyone wishing to gamble at their game and therefore compete against each other on fees, as well as on the reputation of the issuers they choose to work with.

Ignoring dealer and issuer fees, the type of gambling game we discuss essentially consists of collecting s currency units from each of n gamblers and then handing over the whole $s \times n$ amount as prize money to the designated winner, chosen randomly from the n gamblers with uniform probability. This would be a game in which the dealer never makes a profit or a loss. Instead, since there are fees, it is a game in which the dealer is guaranteed to make a small profit at each game run, regardless of who wins—as opposed to a game such as roulette where the dealer has an advantage in the long term but may lose out on individual game runs.

One may wonder why any rational gambler would ever take part in such a game, since the expected return is always less (because of dealer and issuer fees) than the payment required to play. Just so that you can follow the paper without worrying about this point, we refer you to the classic work by Markowitz [8] that explains such chance-taking behaviour through the non-linearity of the function linking wealth to its utility: when we can't afford to lose, we prefer \$1,000,000 with certainty instead of a 1-in-10 chance to win \$10,000,000; but, for much smaller amounts, we may prefer one chance in 10 to win \$10 rather than \$1 with certainty.

4 Cyberdice version 0.1

We now present an initial (flawed) attempt at the Cyberdice protocol; and then discuss how it can be attacked by players who do not follow the rules.

The game starts with the dealer announcing that a game will take place and setting out the rules, the costs, the allowed number of gamblers and the

[2] There is only one dealer per game; but the plural reflects the fact that there may be several independent games.

timescale for the various protocol steps (so that everyone always knows whether they should still wait for someone else's response or not).

This first message, in common with all messages in the protocol, is signed by its sender (in this case the dealer), contains a unique randomly chosen nonce and clearly identifies message type and source/destination. These precautions will ensure that messages from another game or another protocol stage cannot cause any confusion. Viz: we assume throughout this paper that all the usual hygiene precautions are in place, and so we need not worry about replay attacks, duplication of messages, or the use of one message in place of another. Our only concern is attacks upon the high-level design of the protocol.

The gamblers who wish to take part then correspond with an issuer of their choice[3] to purchase a bit-string that represents their game joining fee. This bit string is tied to the particular game and represents an irrevocable commitment by the issuer to pay the fee to the game's winner (or, if it can be shown that the game did not proceed, to reimburse the gambler who deposited that fee). The gamblers send these bit-strings to the dealer asking to take part in the game, along with a commitment to the random number for their dice roll (e.g.: they publish $h(r||x)$, a cryptographic hash of their random dice roll number r concatenated with a nonce x).

The dealer then selects the gamblers for the game, using any criteria she wishes (she may not trust particular issuers to pay up, or she may refuse particular gamblers for personal reasons). Having made the selection, she publishes the list of gamblers who will take part in the game. Since the random numbers chosen by the gamblers are not known to the dealer (they were only committed to), the dealer cannot predict the result of the game as a function of the subset of gamblers she chooses.

The gamblers who were not selected can now recover their money from their respective issuers (by showing the selection message that indicates they will not play in the game). If the dealer fails to make a selection by a given time (which was specified in the original announcement of the game) then all of the gamblers will be entitled to get their money back.[4]

The gamblers now reveal their random numbers (to which they committed earlier) and these values are combined to determine the winner. Just picking the highest value would clearly not work, as dishonest players would choose the highest possible value instead of a random one. We might, if they were all unsigned 32-bit values, add them together modulo 2^{32} and designate as winner the highest number smaller than the total. Alternatively, we might hash each gambler's random number concatenated with the total, and then pick the highest

[3] Nothing prevents several gamblers from choosing the same issuer. In other words there is a one-to-one mapping from the set of players to the set of issuers, but this mapping is not invertible in general.

[4] We will need to describe the mechanism by which we prove this negative, but since this protocol still has much more significant flaws, we will fix those first and come back to this point later.

result. The winner will be able to collect her winnings by visiting each issuer in turn, with the relevant signed messages as proof of her success.[5]

4.1 Timing attacks on Cyberdice 0.1

When the gamblers reveal their random values in turn, the last gambler to reveal will know whether she won or lost before she makes her announcement. If she won, fine; but, if she lost, then there is little incentive—apart from her honesty and sense of fair play—for her to bother announcing the random value, and the game will be incomplete.

The DOS problem can be fixed by adding a further timeout, preset in the initial game announcement message. If a gambler does not reveal their random value within the appropriate time, then she is excluded, and the total (or the hash value) is calculated without her contribution playing any part.

Unfortunately, if gamblers collude, there is still an advantage to be gained from failing to reveal one's random values. For simplicity we will just consider the case where two dishonest gamblers collude, and they both arrange to play 'last'; but the attack is trivially extended to a collusion among the last k gamblers. If they both fail to reveal their random values then they lose; but they now have the choice of revealing one or the other value, or both—and if one of these three alternatives makes one of them the winner they will share the spoils; in other words, they cannot *guarantee* a win, but they can materially improve their chances.

The collusion might not even be pre-arranged. The last gambler can calculate which of two other gamblers will win depending on whether she sends the final message or not. Either or both of these gamblers can be contacted (quickly, as the timeout approaches) with an offer to split the proceeds by guaranteeing a win.

5 Why is online gambling really, really hard?

We can now see that the difficulty with our protocol is that we need to construct a random result from the random contributions of all the gamblers; but if gamblers collude they can have some influence over the result by choosing whether or not to reveal what their contribution was.

We cannot easily fix this problem by fining the gamblers for disrupting the game. The only way to ensure that a gambler does not withhold their random value would be to ensure that they would lose more than they could possibly win—viz: they would have to escrow an amount equal to the total amount being wagered. This might be an acceptable solution where four people were wagering a tenner each, but would make it impossible to run games where a million gamblers were wagering a dollar each—even millionaires might not be prepared to risk their money when an unexpected network outage could see them fined.

[5] Note that at this stage we have not yet provided a mechanism to guarantee that the winner will be unique. See Section 7 for that.

We cannot fix this problem by having the gamblers announce their random values instead of merely making a commitment—the dealer could then rig the game at the point at which the selection of gamblers is made by picking a combination that meant a particularly favoured gambler was the winner.

We could perhaps have the gamblers announce the values encrypted with a 40-bit-key cipher or some other easy-to-break encryption. If they fail to reveal the values then the other gamblers mount a brute-force attack to discover the encryption key. This is not especially elegant, and it would be unwise to play against the NSA—who might be able to decrypt the values at an early enough stage for them to mount a collusion attack. It would be good to have access to a 'nobody will be able to decrypt this message before time t' primitive, but this is known to be a hard problem if one cannot resort to trusted third parties or tamper-resistant hardware.

6 Cyberdice 0.9

Let's skip a few more vulnerable revisions of the protocol and jump forward to one that provides the properties we need. Some readers may feel that our solution is a little bit like cheating: we shall make the assumption that the issuers who provide digital currency for the game are honest. We shall use them indirectly to provide a final randomised stirring of the pool of random values, so as to designate a winner fairly.

We run the protocol exactly as in Cyberdice 0.1 presented above, up until the point at which the gamblers reveal their random values. At this stage, the dealer constructs and signs a message which contains these random values.[6] The signed message is then submitted to each of the issuers used by the gamblers, and they also sign it.[7] The random pool which will be used to decide the winner is created from the values of the signatures which the issuers make.

The game can be seen to be fair, in that it is well-known (albeit possibly hard to prove) that signatures made with high quality cryptographic primitives are random. If this isn't believed to be true of signatures in general, then placing their values into a canonical order and then calculating a cryptographic hash of this concatenation will provide an 'even more random' value.

Essentially, we have finessed away the 'last gamblers can quit when they see that they are losing' problem by asserting that the need of the issuers to preserve their reputation prevents them from reneging on their obligation (paid for in the fees they charge for their bit strings) to provide a signature. Now the last gambler can't yet *see* whether she will win or lose when she is still in a position to quit, and the last issuer (who can see it) is bound not to quit for the reasons above.

We have already given issuers special properties that we do not ascribe to gamblers: we assume that they will act honestly when presented with proof of a

[6] For hygiene reasons, the message might include all previous messages as well, in a canonical order, so as to be sure that no relevant state is being omitted.

[7] This operation is linear in the number of issuers involved, which is bounded by the number of players.

win; we assume that messages to or from them can never be blocked; we assume that they keep an indelible record of the message they have signed; and so on. Assuming that no issuer is so beholden to a gambler that they will withhold their signature does not seem all that unreasonable a stretch. However, we will return to the honesty of issuers in Section 8.

7 Cyberdice 1.0

Now that we have effectively chosen to use the signatures of the issuers as a source of randomness within the game, we can simplify the protocol by eliminating the gamblers' dice rolls. Doing so also finally guarantees the uniqueness of the winner. We present this latest refinement pretty much from scratch (so that it can be read and attacked independently of what led to it) and in somewhat greater detail than we have bothered with so far.

7.1 The principals

The principals taking part in each game are a *dealer* (who neither wins nor loses, but always makes a profit by collecting a fee from the gamblers, independent of the outcome), several *gamblers* (who may win or lose; always pay fees to the dealer and to their chosen issuer in order to play, whether they win or lose) and several *issuers* (who don't take part in the gambling process; they hold money in escrow for a fee and turn it into bit strings that can be later redeemed by presenting other, related bit strings with certain properties). Issuers will also sign messages upon request, under certain conditions.

The main security aim of the protocol is to ensure that, if issuers are honest, a gambler will win or lose with the same probability regardless of whether the other gamblers and the dealer play honestly or dishonestly (they might even all collude against one honest participant).

The protocol is not resistant to dishonest behaviour by issuers; it tries however to offer some robustness by allowing participants the freedom to choose issuers they trust: the dealer specifies up front which issuers are acceptable for the gamblers to use, while the gamblers can see both this list of allowed issuers and the dealer's own choice of issuer before deciding whether to participate.

A gambler pays her chosen issuer a sum consisting of three components: stake s, dealer's fee f_d, gambler's issuer's fee f_{gi}. If she wins, she redeems the prize of $n \times s$ from the dealer's issuer, where n is the number of players who played. The dealer pays her chosen issuer a sum consisting of two components: the maximum possible prize money, that is $n_{max} \times s$, where n_{max} is the maximum number of players that the dealer will admit to the game, and the dealer's issuer's fee f_{di}. At the end of the game, the dealer collects the leftover money $(n_{max} - n) \times s$ from her issuer and, separately, the stakes of all the gamblers from the gamblers' own issuers.

A gambler cannot be cheated by any combination of the other players (gamblers and dealer) if issuers are honest, but could be cheated if issuers are dishon-

est.[8] Similarly the dealer cannot be cheated by any combination of the gamblers if issuers are honest, but can be cheated if issuers are dishonest. An issuer, instead, cannot be cheated by any other participant,[9] whether issuer, dealer or gambler.

Issuers all provide the same service, so they compete against each other on reputation and fees: those who have a good reputation can afford to charge a higher fee. Each issuer has what we shall call a blog—a facility that lets her publish timestamped and signed messages on her web site to an append-only, integrity-protected log. Issuers cannot 'rewrite history' because they must sign the messages they post on their own blog. If an issuer altered and re-signed an earlier post, any reader who kept a copy of the previous version of that post could expose the issuer's tampering, causing a loss of reputation for the issuer and therefore an indirect financial loss.[10] We assume that issuers are sufficiently powerful that communication channels to and from them cannot be cut off by denial of service attacks; in other words it is always possible for anyone to talk to an issuer and to read the blog of an issuer.

7.2 Starting a game

To start a game, the dealer prepares a message of type M_0 ('invitation') that describes the game parameters.[11] These parameters are: the dealer's fee f_d (the 'tax' that the dealer collects from each admitted gambler, forming the dealer's profit), the stake s (the amount that each gambler 'puts on the table' to form the prize money that the winner will eventually receive), the identity of the dealer's issuer (the issuer who will hold the prize money deposited by the dealer and who will pay it out to the winner), the maximum number of gamblers n_{max}, the list of acceptable issuers for the gamblers to use, the deadline t_2 by which would-be gamblers must gamble, the deadline t_5 by which the dealer promises to publish

[8] Issuers must be honest not just about handling money (as is obviously the case for the dealer's issuer, who could refuse to pay out the prize, and the player's own issuer, who could pocket the stake money and not blog (q.v.) the gambler's stake), but also about honouring their contract in terms of signing follow-up messages by a given deadline as agreed. Failing that, it is also possible for a gambler to be defrauded by the action or inaction of another gambler's issuer.

[9] Under the obvious baseline assumptions that signatures can't be forged etc etc.

[10] Conversely, an attacker who were capable of forging the issuer's signature could also frame the issuer by pretending she tampered with her own blog—although an adversary with the ability to forge the issuer's signature, which is outside our threat model, could probably use it for much more profitable attacks.

[11] We continue to assume, as we have done elsewhere in the paper, that all appropriate hygiene precautions are taken to ensure that messages are labelled, and contain appropriate randomly chosen nonces, typically in the form of the ephemeral public keys of the players, so that the protocol is immune from low-level attacks such as replays and the introduction of messages from other runs of the protocol. We also assume that participants always perform all the applicable consistency checks (including verification of signatures) on any messages that they receive, so that simple forgeries will be immediately detected.

the 'game seal' message M_5, as well as some more deadlines necessary to ensure completion of the final payback phase in a finite time (see appendix). The M_0 message also contains the ephemeral public key of the dealer,[12] which doubles as a unique identifier for this game.

The dealer sends this message to her chosen issuer, together with the dealer's issuer's fee f_{di} and enough currency to cover the maximum prize that the game might award: that is to say $n_{max} \times s$. The dealer's issuer, having accepted the payment, timestamps and signs the message (yielding M_1, 'certified invitation') and publishes it on her blog.

Potential gamblers see the certified invitation. In order to play, a gambler must choose an issuer among the ones deemed acceptable by the dealer and send to that issuer, together with an amount of money equal to the issuer's fee f_{gi} plus the dealer's fee f_d plus the game stake s, a message of type M_2 ('stake') containing the following fields: the M_1 to which the gambler is responding, the ephemeral public key of the gambler and the gambler's signature.

The gambler's issuer timestamps, signs and blogs the gambler's message, yielding M_3 ('certified stake').

Then the gambler's issuer also forwards this M_3 to the dealer's issuer. The dealer's issuer in turn timestamps, signs and blogs M_3 to yield an M_4 (a 'doubly certified stake').

Missing deadlines Whenever the dealer misses a deadline, evidence for which is available in the dealer's issuer's blog, all the players get fully reimbursed, fees included, and all the issuers get back (out of the money that was originally left in escrow by the dealer) the fees that they had to refund to the players—so that neither gamblers nor issuers lose out if the dealer fails to perform. The remainder of the money put in escrow by the dealer is then returned to the dealer. So long as the stake s exceeds the gambler's issuer's fee f_{gi}, the money left in escrow by the dealer ($n_{max} \times s + f_{di}$) will clearly be sufficient to reimburse all the issuers, in reason of one f_{gi} per bet; whereas, to reimburse the gamblers, the issuers simply return the same money that the gamblers originally paid, namely $s + f_d + f_{gi}$ per bet.

Meaning of signatures What is the meaning of an issuer's signature on a message? Does it imply that the issuer performed all possible sanity checks on it (e.g.: 'the M_3 refers to a game for which I published the certified invitation M_1 in my blog, there is a good signature on it from some issuer, the issuer was cited as acceptable in the M_0 invitation', etc etc)? Possibly, but not necessarily; a minimalistic and equally valid alternative would be for the issuer to perform essentially no checks and for the signature to mean simply 'I got paid to notarize that I received this string so here is my timestamped certification of it, but I didn't even read it—it could be dirty jokes in ancient Babylonian for all I care' and leave all responsibility for the correctness of the string to the entity who

[12] A new key pair is chosen by each player for each game.

brings the message for signing. There is a trade-off between the two goals of detecting malformed or fraudulent messages as early as possible vs. involving the issuers in as few aspects as possible of the gambling game.

7.3 Deciding on the winner

The dealer monitors the blog of her own issuer for any M_4 that refers to her M_1. She accumulates them in a list until she reaches the stated maximum number of gamblers or, in any case, until she (almost) reaches the t_5 deadline. She then chooses the participants who will take part in the game, using whatever criteria she wishes. She forms a message of type M_5 ('game seal') by signing the concatenation of M_1 with the list of the M_4 of all the chosen participants in order of timestamp. She then submits M_5 to her issuer, who timestamps, signs and blogs it as M_6 ('certified game seal'). If the dealer fails to send her M_5 to her issuer before t_5, the issuer still signs and blogs an 'empty' M_6 containing the relevant M_1 and a flag signalling that no M_5 was received for that game by the deadline. In such a case the game defined by that M_1 is declared void: all gamblers who have an M_4 listed in the dealer's issuer's blog are entitled to get a refund of their stake and of their fees from their own issuer.[13] If instead the dealer sends an M_5 by t_5, then the game proceeds. All the gamblers who submitted an M_2 but are not selected in the M_6 are entitled to a refund of their stake (but not their fee) from their issuer: there is no difficulty for a rejected gambler in proving this entitlement to her issuer, merely by exhibiting both the M_3 and the M_6.

At this point, assuming that the game is proceeding, the M_6 is passed around all the participating gamblers' issuers in turn, in the order in which they were listed in the M_0, and each one of them signs and blogs the bundle consisting of the M_6 with all the signatures collected so far.[14] Once all the issuers involved have signed it (just once each), the bundle comes back to the dealer's issuer as M_7 ('multisigned game seal'). The reason for this round of signatures will be explained after we describe how the winner is selected.

Anyone with access to M_7 can now determine the winner by processing it with the following deterministic algorithm. If n is the number of M_4 messages in M_7, we hash the M_7 and let $i = h(M_7) \bmod n$. This index $i \in 0 \ldots n-1$ points at one of the M_4 messages in the order in which they are listed in M_7, and the gambler who submitted that message is the winner.

The dealer's issuer executes this algorithm, appends the computed i of the winner to the M_7, and then signs and blogs the lot as M_8 ('winner announcement'). The outcome of the game is thereby published and anyone can verify

[13] The money escrowed by the defaulting dealer and held by the dealer's issuer is divided among all the gambler's issuers involved, proportionally to the number of M_3 issued by each, to cover the loss of fees. If any is left over after reimbursing all of the gamblers' issuers' fees, which depends on the number of players who chose to gamble, then it may be collected by the dealer.

[14] Note that we are not assigning a message number to the intermediate versions of the bundle. Doing so would only add gratuitous complication since the number of signers varies from game to game.

that the selection of the winner happened fairly, according to the prescribed rules.

Why do we require all the issuers to sign the bundle in turn? Imagine if none of them signed it, and the message to be hashed were the M_5 supplied by the dealer. Then the dealer could privately try various subsets of gamblers to accept, checking each time who comes out as the winner, and eventually committing to the selection that favoured specific colluding players (or even 'sock-puppets', i.e. fake identities, of the dealer herself). In a previous version of the protocol we blocked this by computing the hash on M_6, i.e. on the version of M_5 signed by the dealer's issuer; but this setup is still subject to the same fraud if dealer and issuer collude. Since the dealer himself could be a sock-puppet of the issuer (!), this is not satisfactory, especially as the fraud is undetectable by someone observing the logs and therefore would not be reflected in the issuer's reputation. By requesting the signatures of all issuers involved, in a predetermined order, this fraud is eliminated unless *all issuers* collude—in which case anyone playing might as well go home.

We further specify that the underlying signature scheme must be completely deterministic, in the sense that once the public key is fixed there is only one possible valid signature for any given message. Failing that, it might be possible for the last signer to fiddle with the signature until the hash pointed at the desired winner.

7.4 Delivering the money

To claim her prize, the winning gambler goes to the dealer's issuer and requests the money by exhibiting an M_{10} ('prize claim'). Logically, this M_{10} need only contain a *reference* to the game, since everything necessary is blogged at the dealer's issuer, but for good measure we'll instead also include in it the whole M_8.

Given M_8, the dealer's issuer (like everyone else) can read out who the winner is; so she must now make sure that she is handing over the prize money to the actual winner. The gambler proves that she is the winner by signing with her secret key a challenge supplied by the dealer's issuer. The 'prize challenge' M_{11} can be seen as a receipt for the prize amount, which the dealer is asking the winner to sign. If the gambler returns it with a good signature (M_{12}, 'prize response'), then the dealer's issuer hands over the prize money. Note that the submission of the prize claim M_{10} must happen within another deadline, t_{10} that was also mentioned in the original M_0. The dealer's issuer in turn timestamps, signs and blogs the winner's response as M_{13} ('prize receipt'), to let everyone know that the winner confirmed that she received the money (or, alternatively, that nobody came forward to claim the prize).

We emphasize that it would be possible for the dealer's issuer to defraud the winner here (the winner's prize response, which is in effect later treated as proof that the winner was paid, is signed by the winner just *before* the money is handed over), which is why the issuer is 'trusted'—using Morris's definition of a trusted entity as one that can violate your security policy. If either the payment

or the receipt has to happen first, in the window between the two events the party who moved first is exposed if the other quits. We trust the issuer more than the player so we put the heavier burden on the player, but we highlight the importance of the requirement of atomicity for any 'transducer' subprotocol between an issuer and a player in which a conversion between money and a bit string takes place: this happens now, at the end of the game between winner and dealer's issuer, but also between dealer and issuers, and also at the start of the game when dealer and gamblers escrow their money. In all these cases atomicity is required, and in all these cases we give the advantage to the issuer. All these 'transducer' subprotocols are marked with an 'atomic' bracket in the appendix.

For subprotocols where an issuer pays a player rather than vice versa (M_{10}–M_{13}, M_{20}–M_{23} and M_{30}–M_{33}, collectively referred to as $M_{?0}$–$M_{?3}$), we require the issuer to include low-level evidence of the payment having taken place (think of some kind of bank transfer reference) in the blogged 'receipt' message $M_{?3}$. That way, if a dishonest issuer obtains a signed $M_{?2}$ from the player but fraudulently doesn't pay her, then evidence of such non-payment is accessible to all in that issuer's blog.

Finally, within yet another deadline t_{20}, the dealer must visit her own issuer and claim any leftover money (e.g. if she deposited money for a 20 person game but only 14 played) and also visit each of the gamblers' issuers to collect the gamblers' game stakes, using a challenge-response subprotocol very similar to the one used above by the winner to prove that she was the dealer and exhibiting M_{13} as evidence that the game concluded properly and the legitimate winner got paid.

As an aside, the full game definition should also include clear rules about who gets to keep the escrowed money if those entitled to it fail to claim it by the deadline.

7.5 Discussion

Assuming that all issuers are honest, the protocol ensures that neither the dealer nor any gambler can defraud the other players. Crucially, this is achieved without specifically involving the issuers in the gambling process: the issuers only honour their pre-agreed contracts about issuing bit strings for money and later converting other bit strings back into money (provided that those bit strings verify certain pre-agreed properties). Conceivably, the issuers could offer this same service for other purposes than allowing their customers to play Cyberdice, which makes their role independent of the gambling aspect and, therefore, potentially legitimate even in jurisdictions where gambling might be illegal. This, in turn, leaves—in theory—some scope for official regulation of issuers and statutory protection of customers against misbehaviour by an issuer.

If some issuers are dishonest, players are vulnerable. If a gambler left money in escrow with a dishonest issuer, the dealer risks not receiving that money (which legitimately belongs to her, since it repays an equivalent amount she deposited with her own issuer to be given out as prize money). Similarly, if the dealer chose a dishonest issuer, the winner risks not receiving her legitimately

earned prize. At least in both of these cases it is easy to convince any third party, including an arbitrator, that the issuer misbehaved, simply by exhibiting the messages signed by the various parties. This can be used to accumulate reputation credits (both positive and negative) for the issuers. Since we explicitly allow gamblers and dealer to accept or refuse to take part in a Cyberdice game on the basis of the issuers involved, we at least have a mechanism to protect players against known-bad issuers. Although it would be possible for an issuer to suddenly turn bad despite an unblemished reputation history, our framework tries to prevent this from being advantageous for the issuer: while players are essentially anonymous and short-lived (they all use ephemeral keys), issuers by contrast are long-lived and owe all their business (potentially including business not related to Cyberdice) to their good reputation. So long as cheating by an issuer is detectable and worth much less than the rest of the business that a reputable issuer might legitimately obtain, then incentives are properly aligned to deter issuers from cheating.

Much more worrying is the possibility of issuers committing frauds that the protocol cannot detect, such as the one mentioned above when we explained what might happen if we didn't ask all issuers to sign the M_6. Even though it is hard to defend against frauds by entities we are forced to trust, as the Morris quote cited earlier reminds us, we would like the protocol to at least allow *detection* of frauds perpetrated by a single crooked issuer in presence of other honest ones.[15] We consider it acceptable to be vulnerable to undetectable frauds in the case where all issuers collude against the players, especially given all the subtle issues related to timestamping that we are not examining for lack of space.[16]

8 Related work

The game of Cyberdice is essentially a secure multi-party computation, a problem for which there is a rich literature, dating back at least to the Byzantine generals problem [7]. However, many of the published protocols implicitly assume that the participants comply with the rules and will send all of the messages required of them. For example, in Yao's solution [10] to the Millionaire Problem (compute the Boolean value 'x greater than y', knowing x and without learning y) his Alice learns the result the computation first and must send the final message to Bob to allow him to obtain the same information.

The game portion of Cyberdice, excluding the crucial payment issues, is the multi-party computation of a random number, clearly connected to the widely studied problem of distributed coin flipping [1,2].

[15] This suggests that players should not all use the same issuer, otherwise it would be too easy for 'all issuers' (i.e. just the one) to collude against the players. There is robustness in diversification.

[16] The authoritative clock for the Cyberdice timestamps ought to be, for consistency, that of the dealer's issuer. What attacks could a malicious dealer's issuer perform by fraudulently manipulating that clock? Could we build any defenses around the hypothesis that at least some of the other issuers may have the correct time throughout? Etc.

In multi-party computation the requirement that it be hard to construct a commitment knowing other players' commitments (but not the values) is known as *non-malleability*, a term put forward by Dolev et al. [3]. Combining this with a robustness requirement (that players cannot give up in the middle of the protocol) gives a property called *independence*. Various protocols [6,9] for solving the problem have been proposed but they require $\mathcal{O}(N)$ messages and $\mathcal{O}(N^2)$ computations and they assume that only a proportion (typically at most 1/2 or 1/3) of the players are cheating. In Gennaro's scheme [6], verifiable secret sharing (VSS) is used by each participant to make their commitments. A (t, n) threshold scheme is used to verify (in a zero-knowledge manner) that there is no cheating at this stage. If at a second, reveal, stage any participant fails to take part then the holder of the shares of their secret can reconstruct their key and reveal the value they committed to.

Faust et al. have improved this in v-SimCast [5] by using Gennaro's scheme for commitment combined with Rabin's idea of backing up secret keys using VSS [9]. The scheme does not require any zero-knowledge proofs and is particularly efficient when multiple rounds occur with the same participants, because the VSS operation need only be performed once. The scheme is secure provided that half or more of the participants are honest.

Faust's scheme is therefore inappropriate for our threat model for cheating players, in that we have a constantly changing population (so the efficiency of running multiple rounds is absent) and because we wish to enable one honest gambler to take part even if everyone else is crooked. However, it is much more appropriate for solving the issues we raise in Sections 7.3 and 7.4 to enable us to deal with a small proportion of issuers failing to live up to their responsibilities.

Finally, the fear that in cyberspace everyone else may be out to get you has clear parallels with Douceur's description of the Sybil Attack [4]—where an attacker attempts to control a peer-to-peer system by pretending to be multiple independent identities which we previously referred to as 'sock-puppets'. Douceur shows that, apart from making unrealistic assumptions about resource parity and coordination, the only way that such attacks can be prevented is by introducing a centralized certifying authority that attests to the uniqueness of the participants. Although Cyberdice is much more permissive and does not preclude players (including dealers) controlling multiple identities, Douceur's principle has some resonance with our need to rely upon the good behaviour of the issuers.

9 Conclusions

We presented a mechanism to allow gamblers safely to 'put money on the table' in cyberspace, based on issuers holding money in escrow in exchange for bit strings that can later be redeemed when certain conditions are met. Interestingly, issuers offer a legitimate service that is totally independent of the gambling game.

We presented a protocol that selects a winner fairly (i.e. randomly) even if all the gamblers and the dealer are dishonest, and even if all players but one collude against the remaining one.

Although this result might at first appear to exceed the theoretical boundaries established in the literature for secure multi-party computation, in that it reaches a fair outcome even if *none* of the players is honest, it does so by relying on external entities, the issuers, whom the players must to some extent trust. We have strived to keep the issuers as far away as possible from the gambling process and to ensure that their behaviour can be audited but it is still possible for them to defraud the players undetectably if they all collude.

We consider this work an interesting exploration of the issues surrounding a completely adversarial multi-party computation, in which principals not only might send forged messages but might even stop playing altogether as soon as they notice that they are losing.

Acknowledgements and history

For the serendipitous stimulus that led to this work we are grateful to Frank's graduate student Bogdan Roman, who was investigating efficient MAC-layer techniques for implementing leader election protocols in ad-hoc wireless networks. Frank suggested that he consider the case in which participants cheated instead of cooperating, but this alternative viewpoint seemed less interesting for the student, whose focus was on performance. Frank attempted to caricature the scenario as a game, to make the issues clearer, and subsequently—by then on his own—worked on a financial framework and protocol to implement and play the game. In July 2007 he gave two work-in-progress talks on Cyberdice to the Security Group at Cambridge, where he received quality feedback from several members including at least Richard Clayton, Matt Johnson, Markus Kuhn, Piotr Zieliński, Ross Anderson, Mike Bond and, by mail, Mark Lomas. Frank then teamed up with Richard, who had offered some of the most insightful comments and attacks, to fix further vulnerabilities and consolidate the work into the version presented at the Protocols workshop in April 2008. George Danezis and Emilia Käsper commented on a draft and gave useful advice on related work. Any remaining vulnerabilities are of course still the responsibility of the authors. The sequence of numbered versions listed in the paper is a fictionalised simplification of the much longer litany of attack-defense-counterattack revisions that Cyberdice actually went through.

References

1. A. Z. Broder and D. Dolev: Flipping Coins In Many Pockets (Byzantine Agreement On Uniformly Random Values). In: 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 157–170.
2. M. Ben-Or and N. Linial: Collective coin flipping. In "Randomness and Computation" (S. Micali, Ed.) Academic Press, New York, 1989, pp. 91-115.

3. D. Dolev, C. Dwork and M. Naor: Non-malleable cryptography. In: Annual ACM Symposium on Theory of Computing (STOC '91), 1991, pp. 542–552.
4. J. R. Douceur: The Sybil Attack. In: P. Druschel, F. Kaashoek, A. Rowstron (Eds.): Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge MA, USA, March 7–8, 2002, LNCS 2429, 2002, Springer, pp. 251–260.
5. S. Faust, E. Käsper and S. Lucks: Efficient Simultaneous Broadcast. In: R. Kramer (Ed.): Proceedings of the practice and theory in public key cryptography, 11th international conference on public key cryptography, PKC'08, Barcelona, Spain, March 9–12, 2008, LNCS 4939, 2008, Springer, pp. 180–196.
6. R. Gennaro: A protocol to achieve independence in constant rounds. IEEE Transactions on Parallel Distribution Systems, 11(7), 2000, pp. 636–647.
7. L. Lamport, R. Shostak and M. Pease: The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), July 1982, pp. 382–401.
8. H. Markowitz: The Utility of Wealth. Journal of political economy, LX(2), 1952, pp. 151–158.
9. T. Rabin: A simplified approach to threshold and proactive RSA. In: Advances in Cryptology—Crypto 98, 1998, pp. 89–104.
10. A. C. Yao: Protocols for Secure Computation. Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 1982, pp. 160–164.

Appendix A Timeline of messages in Cyberdice 1.0

Dealer **sends** M_0 (invitation) **to** Dealer's issuer
Dealer **pays** prize MONEY and fees **to** Dealer's issuer
Dealer's issuer **blogs** M_1 (certified invitation)
} atomic

Gambler **sends** M_2 (stake) **to** Gambler's issuer
Gambler **pays** stake MONEY and fees **to** Gambler's issuer
Gambler's issuer **blogs** M_3 (certified stake)
} atomic

Gambler's issuer **sends** M_3 (certified stake) **to** Dealer's issuer
Dealer's issuer **blogs** M_4 (doubly certified stake)
} repeated for each would-be gambler

Deadline t_2: Gamblers must submit M_2 by this time.
Issuers guarantee to blog and forward the corresponding M_3 shortly afterwards.

Dealer **sends** M_5 (game seal) **to** Dealer's issuer
Dealer's issuer **blogs** M_6 (certified game seal)

Deadline t_5: Dealer must submit M_5 by this time.
Issuer guarantees to blog M_6 (full or empty) shortly afterwards.

All issuers accumulate signatures on M_6 in order, yielding M_7 (multisigned game seal)
Dealer's issuer **blogs** M_8 (winner announcement)

Winner **sends** M_{10} (prize claim) **to** Dealer's issuer
Dealer's issuer **sends** M_{11} (prize challenge) **to** Winner
Winner **sends** M_{12} (prize response) **to** Dealer's issuer
Dealer's issuer **pays** prize MONEY **to** Winner
Dealer's issuer **blogs** M_{13} (prize receipt)
} atomic

Deadline t_{10}: Winner must submit M_{10} by this time.
Issuer guarantees to blog M_{13} (full or empty) shortly afterwards.

Dealer **sends** M_{20} (stake claim) **to** Gambler's issuer
Gambler's issuer **sends** M_{21} (stake challenge) **to** Dealer
Dealer **sends** M_{22} (stake response) **to** Gambler's issuer
Gambler's issuer **pays** stake MONEY **to** Dealer
Gambler's issuer **blogs** M_{23} (stake receipt)
} atomic

} repeated for each gambler listed in M_5

Dealer **sends** M_{30} (leftover claim) **to** Dealer's issuer
Dealer's issuer **sends** M_{31} (leftover challenge) **to** Dealer
Dealer **sends** M_{32} (leftover response) **to** Dealer's issuer
Dealer's issuer **pays** leftover MONEY **to** Dealer
Dealer's issuer **blogs** M_{33} (leftover receipt)
} atomic

Deadline t_{20}: Dealer must submit *all* the M_{20} and M_{30} by this time.
Issuers guarantee to blog M_{23} or M_{33} (full or empty) shortly afterwards.
