A Real World Application of Secure Multi-Party Computations
(Duplicate Bridge for Cheapskates)

Matthew Johnson and Ralph Owen
matthew.johnson@cl.cam.ac.uk, rho21@cam.ac.uk

University of Cambridge

Abstract. This paper presents a use of secure multi-party computations in duplicate bridge. A two-person computation is performed by dealing cards in a specific permutation in order to create a deal which has been pre-determined by computer. Statistical analysis is used to demonstrate that little information about the resultant hands can be derived by the participants of the computation. Experimental evidence for failure-rates and time taken to perform the operation is also presented.

1 Bridge

To turn bridge from a game involving a lot of chance into one of skill a duplication system is used such that at least two different sets of people play with identical cards. Your score is then not the number of points you won or lost on that hand but the difference between your score and that of the other people playing with the same cards. Thus being dealt abstractly better cards does not automatically give you a better result.

There are two schemes for achieving this. The naïve approach would be to create several decks of cards sorted identically. This can be avoided using a schedule of playing multiple hands in which each set of cards are passed round between the people playing them in order. Thus, a random deal can happen at the start of the night and as long as the hands are preserved between plays the duplication is achieved.

There are circumstances where this is still not sufficient. At very large events you may need multiple people to play the same hand simultaneously. For these the duplimate system was invented. All the decks of cards are marked with bar-codes so that when fed into a duplimate machine it can deal them into the right hands automatically. A computer then randomly generates sets of cards for the duplimate machine to produce.

Aside from making it possible to run large events, having computer generated hands has a number of benefits which it would also be nice to have in smaller events. Computer generated hands tend to produce more random, and hence more interesting, arrangements of the cards. It is also the custom in clubs which can afford a duplimate machine to produce a record of the hands which can be taken away and studied later. Recently, this is also accompanied by an analysis of the hands giving the contracts it should be possible to make with those cards to aid the players' reviews. There are also some competitions in which the same boards are played in clubs across the country in order to get comparisons between the players in different clubs.

Generating random hands on a computer and generating records of the hands are both simple tasks. Even analysing hands for the best contract is a solved problem, with several pieces of commercial [1] and open source [10] software implementing it and some papers [2] writing about it.

Without a duplimate machine, however, turning these into hands which can be played with actual cards is a long, tedious affair, which necessitates the person creating the hands observing them in the process and hence not able to play them. While in large events the Tournament Director is not a player, in most clubs and smaller events they will be one of the players.

2 Multi-Party Computations

Ultimately we can see the root of multi-party computations in Diffie and Hellman's seminal work introducing public key cryptography [4]. It wasn't until 1979, however, that Shamir [11] introduced the original use

of multi-party computations, that of secret sharing. The Shamir scheme divides a secret between several different people such that a minimum sized subset of them must perform a shared calculation to recover the original secret. This threshold scheme was created using interpolation over a prime field.

In 1982 Yao [12] presented a general solution for any m parties calculating any function f so that the input from each party is kept secret from the others. Yao's solution used public key cryptography. A similar result using zero-knowledge protocols was achieved by Goldreich et al [5] in 1987. These was confirmed with a stronger result by Chaum et al [3] in 1988. Chaum allows the participants secure communication channels and as such does not need to rely on the trapdoor functions used in the previous protocols. In practice such channels often exist.

These protocols are, however, not sufficient for the task of duplicating cards securely. They differ in two important respects. Firstly, the computation which is carried out by the parties typically requires a computer to perform whereas in this situation the players must be able to perform it manually.

Secondly, the aim of the protocols in the literature is to produce a public result based on secret information. Here we wish to produce a secret result. Ordinarily the model of communication used between the players results in any intermediate state which is passed between participants being known to them, which means the final result is always known by at least the last participant.

However, because these operations are being performed manually on a deck of cards it is possible for participants to receive an input, perform an operation on it and send it to the next participant without being aware of any intermediate state, as long as the cards remain face down. Thus while the first participant may know the initial state (and hence the state after their operation) it is possible to produce a scheme in which the state produced by subsequent participants (and hence the result) is unknown to all of them.

This implies a certain level of trust in the participants not to cheat by looking at the cards as they are operating on them, however, this is a level of trust which is already present in many places throughout the game.

3 Faking Duplimate

A number of years ago the notion of dealing sheets was introduced by Handley [7]. These are an alternative to a thorough shuffle before dealing used by several people in Cambridge who find it difficult or tedious to perform a well-randomising shuffle. They work similarly to one time password systems.

The user is given a sheet with a number of permutations of the string 111...222...333...444... which are used once and then crossed out. The permutation is used to deal out a pack of cards which has been given a superficial shuffle to place them in an unknown (but not statistically random) order. The first card from the deck is place in the pile corresponding to the first number in the permutation. The second card is placed in the pile corresponding to the second number, and so on.

Because the deal is following the (randomly generated and equiprobable) permutation and not being dealt in order as is usual the shuffle does not need to provide any statistical randomness. All that is required is that the deck is put into an unknown state to begin with.

This scheme is, however, only good for producing random deals, not pre-computed ones. What is needed is a way to put the deck into a starting position known by the computer and not by the person dealing according to the dealing sheet. At first glance this seems equivalent to our original problem until you realise that the state of the deck may be known to a different player, since if they don't see dealing sheet they can't know how the final deal ends up.

This leads neatly onto multi-party computations. If performing a deal according to a dealing sheet is considered an encryption E under a key given by the dealing sheet k then the composition of two encryptions under different keys $\{\{d\}_{E_{k_1}}\}_{E_{k_2}}$ will result in a deal which is calculable given the original deck d, and the two keys k_1 and k_2, but completely unknown to anyone who knows only two of the three.

Thus, duplication of a pre-computed set of hands is possible without any of the human participants being able to calculate the hands. The computer generates the desired result hands and the two keys, starting from a sorted deck. These keys are given to different people. The person with key 1 performs a dealing sheet operation on the sorted deck, passes the result (combined to a single deck) to the second person, who

then performs a dealing sheet operation with the second key. Neither will be able to gain any knowledge about the result but it will be dealt according to the pre-computed hand.

There are, however, a number of issues with the scheme which has just been described. Firstly there are issues with errors, error detection and error recovery. Because the calculations are performed by humans rather than computers there is a high chance of errors creeping in. Section 3.1 covers options for error detection and recovery and section 6 gives results from a test of the system in the field.

The other, more important, issue is that because each permutation is not a randomly selected element of G_{52} there is still some information which may be learnt by the people performing the operations. An analysis of this along with solutions to, or at least ameliorations of, the problems follows in section 4.

3.1 Error Detection and Recovery

It may be the case that error detection and recovery does not matter. If a fault occurs during the dealing the result will still be a good-quality random deal (modulo an incorrect number of cards in each pile, which can be rectified before any player looks at their cards). The only problem is that it won't match the hand record. If the number of these is sufficiently small then there may be no need to do anything further.

It may well be the case that more than this is required. For error detection there are two schemes which were trialled in this research. The simplest is for one of the players to have a record of what should be in each hand. After they have played each hand the four players at that table check the record against the actual hand to spot differences. This does not allow for any correction since the board has already been played, but small errors such as exchanging a card or a suit can be noted. The hand record can then be amended for these small errors in retrospect.

In the case of large errors where no simple correction to the hand record can be made, the hand record for that deal must just be marked invalid. Players can be given a list of deals for which the hand record is incorrect. For small rates of this sort of error this solution may be acceptable.

In cases where you need the hands to be exact, for example with simultaneous pairs competitions where many clubs play the same hands, any necessary corrections must be made before the first deal. When manually creating the hands for this sort of event similar error detection is needed and for this a common solution is the use of curtain cards. These are a record of what each individual hand within a deal contains which are passed with the cards, one in each pocket of the board. Players at each round (or at least the first) verify that the contents of their hand matches the curtain card.

Curtain cards detect errors before the hand has been played, so correction is possible, but there are issues with leaking information. If two players report they have one incorrect card, it will be obvious to each of them where their incorrect card ends up. In some cases this will not matter, but in others it may change the result of the play.

A solution which doesn't have this problem, but does require a non-player, is to have all the deals checked by a non-player before the event starts. This is still not as tedious as the non-player manually arranging the hands (verifying them is faster than creating them) and all the dealing work can be done by players in parallel.

4 Statistical Analysis

This scheme is designed to provide security against accidental or casual storage of information from the deal, it is not meant to be secure against a serious or organised attack. Therefore, we will only consider simple attacks in this paper, and we will assume that the two parties in the computation will not collude and that they will only use information provided to them during the algorithm (and not cheat in other ways).

4.1 Suit of first card dealt

Problem Consider the knowledge of the location of the first card dealt in the second operation (hereafter P_2).

The initial deck will always be sorted identically, by suits, so that the final 13 cards dealt in P_1 are always the clubs, say. However, the first card dealt in P_2 is the last card dealt to pile 1 in P_1, so this will always be a club, unless no clubs are dealt to pile 1 in P_1.

No club will be dealt to pile 1 in P_1 with probability

$$\Delta = (39/52 \cdot 38/51 \cdot \ldots \cdot 27/40) \approx 1.27\%.$$

This is, naturally, precisely the chance of holding a club void in a given hand. The problem is that even if the first card in P_2 isn't a club, the hand receiving it might get a club later in the deal.

Without inspection of when in the rest of P_2 cards are dealt to this same hand, we can estimate the probability that this hand does not hold a club as

$$\Delta \cdot (38/51 \cdot 37/50 \cdot \ldots \cdot 27/40) = \Delta^2 \cdot 4/3 \approx 0.022\%.$$

So the hand receiving the first card in P_2 is about 58 times less likely to hold a club void than the *a priori* odds.

Generalisation The last card dealt from P_2 produces an equivalent effect with the suit at the other end of the initial sorting, say spades. Furthermore, the cards at the ends of each pile in P_1 (i.e. dealt in positions 13, 14, 26, 27, 39, 40 in P_2) will have the same effect in either spades or clubs. [The combined effect of looking at all of these at once goes beyond the scope of this paper.]

Lesser inferences can also be drawn from cards in other positions. The second card dealt, for example, provides a much milder implication that the had it is dealt to does not hold a club void.

Solution The most useful solution to this problem is to randomise the initial suit order. With this modification, a single card assignment tells you that a given hand is less likely to hold a void in an unknown suit - entirely worthless.

Initial suit order security The initial suit order can often be determined from a look at a hand in the order it was dealt.

Because in P_1 the suits are dealt in turn, the piles at the end of P_1 will be ordered according to the initial suit order. Thus a look at the order in which cards were dealt to a hand in P_2 is very likely to allow determination of the initial suit order, which undermines the solution to the previous problem.

As this is a sub-problem of the previous problem, there is only a security issue if the person who performed P_2 is first to see the hand in question. Whilst it could be secured by arranging that this doesn't happen, a simpler solution is to add a short shuffle after the hands are dealt.

Final thoughts The combination of several card assignments still might give some information. If you hold a void and received a card from one of the sets of positions $\{1, 14, 27, 40\}$ and $\{13, 26, 39, 52\}$ in P_2 (i.e. the ends of piles), you may be able to conclude that your void suit was less likely to have been at either end of the initial suit order, and so that any other hand receiving a card from the same set(s) of positions as you is less likely to hold a void in each of the suits you don't hold a void in. This is pretty tenuous, however, and unlikely to be relevant at the level of security we are seeking. Anyway, it is relieved by the solution to the next problem.

4.2 Locating high cards

Problem The bottoms of the piles in P_1 will generally contain high cards in the suit first in the initial suit order.

If a hand is seen to receive all four of the cards from positions $\{13, 26, 39, 52\}$ in P_2, it is certain that this hand holds the Ace of this unknown suit, there is a $3/4$ chance that it holds the King of the same suit, a

$3/4 \cdot 1/2 + 1/4 \cdot 3/4 = 9/16$ chance that it holds the Queen, etc. Although the suit is unknown, it may quickly become apparent during the bidding (as there are plenty of bids which show specific cards) or play (when another hand is visible on the table). Potential results of this include:

- During the bidding, guessing that your partner, say, holds the Ace and King of a suit rather than just the Ace he has shown. This may allow a better contract to be reached.
- During the play, locating a missing high card. This is often extremely valuable in deciding how to play the cards, and may result in a contract making where it would have failed, for instance.

Of course, the same applies with the suit at the other end of the initial suit order, except that here we are finding the low cards, which are much less likely to be valuable, especially when it is difficult to determine their suit.

Even if a hand receives fewer than all four of the cards from those positions, useful inferences can be taken, especially when holding some of the missing cards.

Solution A first solution is to modify the number of cards dealt to each pile in P_1. This needs to be done such that the three end points in the middle of the pack are independent of each other in position. However, it is also valuable to avoid having too many cards in any one pile, as this will result in a far higher than usual number of consecutive cards in this pile, which may surprise the person performing P_1 and result in insecure comments about the strangeness of the deal.

In order to make three suitably independent positions, we selected two random numbers r_1, r_2 and used the pile sizes r_1, r_2, $26 - r_1$ and $26 - r_2$. This results in the piles ending at positions r_1, $r_1 + r_2$, $26 + r_2$ and 52. Using even distributions between 10 and 16 for r_1 and r_2 means each pile will have between 10 and 16 cards. The middle end point will be distributed around 26 with a stronger weighting at the centre, but the outer two end points will be uniformly distributed.

This means that the ends of the piles cannot be accurately predicted, with the exception of the cards in positions 1 and 52, so it is much harder to gain useful information here.

Remaining issues Nonetheless, when missing only one high card, it seems plausible to take the inference that the person dealt the last card in P_2 holds it. With more high cards missing, you can infer that this person was dealt one of them, which alters the expected distribution and might be used to choose a better line of play. In any case, this is only likely to be valuable in a slam contract.

One solution to this would be to have the person who performed P_1 (or a third person) perform a defined permutation of the hands after the completion of P_2. This is not ideal as it adds a new source of error to the process, at a guess a comparatively large one.

Also, any suggestions for a better way of distributing the number of cards in the piles in P_1 would be useful.

4.3 Further work

It would be interesting to perform a Monte Carlo simulation to attempt to spot further patterns in the distribution of cards as this might highlight new flaws in the scheme.

5 Pescetti

The system described in this paper has been implemented in the Pescetti PseudoDuplimate software [8]. Pescetti is written in Java and released under the terms of the GNU GPL version 2.

The heart of Pescetti is a random permutation generator based on Knuth's shuffle [9]. This takes an array of 52 elements containing the numbers 1 through 4. For each element it is swapped with a random other element in the array. The random numbers for this are generated by the built-in Java class SecureRandom.

This produces a random permutation of the cards into four piles with statistically even likelihood of any particular distribution.

To generate the desired random deal a random permutation of 13 of each number is generated and this is applied to the sorted deck, generating a perfectly random deal. This is the target hand which the permutations have to generate.

As per section 4.1 an initial permutation of the suits is randomly generated using the Knuth shuffle on an array of the four suits. This generates an initial start deck sorted within each suit, but with the suits in a random order.

Another complete permutation is then randomly generated using the Knuth shuffle. As per section 4.2 this starts with not 13 of each number, but at random between 10 and 16 of each number. This permutation is given as the first permutation for the players to deal, along with the initial permutation of the suits.

This permutation is applied to the deck produced by the initial permutation of the suits to generate the intermediate deck between the two permutations. Pescetti iterates over the intermediate deck, looking up each card in the target deck to produce the second permutation.

Because the first permutation and the target hand are both independently random the second permutation is also independent of the resultant hand. Pescetti generates sheets containing the permutations to be dealt. Each page contains the first permutation for two deals and the second permutation for two deals. A corresponding page contains the opposite combination for those four deals. This allows two dealers to parallelise the production of the final boards.

Pescetti calls out to a double dummy solver written by Bo Haglund [6] to produce hand records containing double-dummy results.

6 Case Study

Pescetti was tested at the Cambridge University Bridge Club in November 2007. This trial was used to discover what problems there were with the human part of the system. and what improvements could be made.

6.1 Time

One of the pre-requisites of this system being acceptable is that it can be completed in a reasonable length of time. The two points we are comparing this with are: shuffling and dealing random boards with no hand record and manually arranging hands into a pre-dealt set of hands. The former is the normal operation for any club without a duplimate machine and should be regarded as an ideal target time. Taking longer may be acceptable given the extras which you get from the system. Getting close to that time would make it a plausible system for use each week.

If the system takes too long for that, but is still faster than manually sorting hands then it may still be usable on the occasions that this is required. There is also the added improvement that you do not need a non-player to setup the boards.

Timing results In the trial there were typically between two and four pairs of people dealing the boards. Each had two or three sets of four boards to deal between them for a total of around twenty four, depending on the movement and number of pairs playing. The first couple of weeks took around twenty five minutes, but this has reduced to around fifteen minutes fairly consistently.

This compares favourably to shuffling and dealing randomly. It is not as good (typically around ten minutes to shuffle and deal), but most people do not spend long enough shuffling the cards to get completely random hands.

At the end of the night every player was asked to sort one board which only took a couple of minutes for the whole set to be sorted ready for dealing in subsequent weeks.

For comparison it takes fifteen to twenty minutes to put a set of boards through a mechanical duplimate machine. The method described here is therefore very comparable in terms of time to other methods of computer dealing.

6.2 Error Rates

Since there are a number of failure modes which cannot be detected during the dealing process some errors will be present in the final deal. High error rates will diminish the usefulness of the process and therefore to be successful we need to show they are sufficiently small.

Error Results For the first week curtain cards were provided for all the hands and players asked to check the first time they were played. It proved impractical to fix the errors during play and so for subsequent weeks they were checked by the director as the boards passed his table.

The error rates did not decline as hoped, ranging from nine on the first night through to six on the final night. However, it was observed that a number of the failures coincided with certain people dealing them. On the one occasion where only two pairs who were experience at using the dealing sheets produced the boards there were only four errors.

The errors varied from single cards or entire suits being swapped, through multiple single-card mistakes to some hands where all or most of the cards were incorrect. None of these can be corrected in play, but some are very easy to correct if a non-player with the desired hand records checks the hands.

Some of the dealing mistakes were caught during the dealing process and caused re-deals. This happened typically once or twice per week. In addition there was usually one board which was discovered at the start of play containing an incorrect number of cards in each hand. This could be discovered during dealing by counting the cards into each hand after dealing, but sorting and re-dealing is quite time consuming.

An interesting type of error is that when an entire suits is swapped. Caused by a failure in the initial ordering of the suits, these errors don't affect the analysis of the hands: the reader need just swap the analysis of the suits as well. It may affect calculation of par scores.

7 Conclusion

This method of producing duplicated boards is at least partially successful. It does not take too long to be feasible and the error-rate is around 15–20%. For applications such as a club night which would otherwise just have random boards it does provide added value and better quality hands. With experience of those using the dealing system it should be possible to keep error rates in the 10–15% region.

In terms of security, the statistical analysis shows that the information leakage to the dealers is very small. It is unlikely that any of the dealers would be able to correlate the information they have without the aid of a computer in order to influence the game in anyway. Certainly it is secure against accidental discovery of information by an otherwise honest party, which is the main goal of this technique.

For applications where this is not feasible it would be possible to have the hands checked by a non-player before the game and re-deal those which fail. This is time consuming, but less than having non-players create all the hands from scratch (a process which is not error-free either). In the absence of a duplimate machine, this is the best available solution when all hands must be accurate.

Acknowledgements

The authors wish to thank all the members of the Cambridge University Bridge Club for their patience when trying out new dealing methods and for those who had useful comments about how to improve the system.

References

1. William Bailey, Evan Goldberg, Bill Ford and George Kucera. "Deep Finesse". `http://www.deepfinesse.com/`.

2. Ming-Sheng Chang. "Building a Fast Double-Dummy Bridge Solver". Tech. Rep. TR1996-725, New York University Computer Science, Aug 1996. http://cs.nyu.edu/web/Research/TechReports/TR1996-725/TR1996-725.pdf.
3. David Chaum, Claude Crépeau and Ivan Damgdrd. "Multiparty Uncoditionally Secure Protocols". In "20th Annual Symposium on Theory of Computing", pp. 11–19. ACM, 1988. http://portal.acm.org/citation.cfm?id=62214.
4. Whitfield Diffie and Martin E Hellman. "New Directions in Cryptography". *IEEE Transactions on Information Theory*, **IT-22**(6):644–654, Nov 1976.
5. Oded Goldriech, Silvio Micali and Avi Wigderson. "How to Play Any Mental Game". In "19th Annual Conference on Theory of Computing", pp. 218–219. ACM, 1987.
6. Bo Haglund. "DDD". http://web.telia.com/~u88910365/.
7. Ben Handley. "Dealing Sheets". Personal Communication.
8. Matthew Johnson. "Pescetti—Pseudo-Duplimate Generator". http://www.matthew.ath.cx/projects/pescetti/.
9. Donald E Knuth. *The Art of Computer Programming volume 2: Seminumerical algorithms*, chap. 3, pp. 124–125. Addison-Wesley, 1969.
10. Bob Richardson. "DDS". http://www.bridge-captain.com/downloadDD.html.
11. Adi Shamir. "How to Share a Secret". *Communications of the ACM*, **22**(11):612–613, Nov 1979. http://portal.acm.org/citation.cfm?id=359176.
12. Andrew C Yao. "Protocols for Secure Computation (extended abstract)". In "23rd Annual Symposium on Foundations of Computer Science", pp. 160–164. IEEE, Nov 1982.