# SquareDeal: Tournament Hand Management Software

*Hans van Staveren*

## Summary

SquareDeal is a piece of software for people that make sets of bridge hands for potentially large tournaments. It uses the well-known program BigDeal for actual dealing, but adds two things:

1. An easy way to make multiple sets of deals for various sessions using consistent naming.
2. A way for participants to check the hands were dealt honestly and without any tricks by the organizer.

Using SquareDeal a specific procedure is needed, in various periods, by organizer and participants. There is no actual need for participants to do anything, but if they want to check the hands after the tournament, at least some of them need to do something even before the tournament. If you only want to use SquareDeal for the first of the objectives, you can ignore all the timing and publishing stuff.

I will describe the periods and for each period I describe some parts of the software and/or procedures relevant to that period. The periods follow each other in time, so I will describe them after the heading T0, T1, etc. The program performs two different functions, separated by the *publish* event (see later).

This document will be released together with the software. The software is written in Perl (about 1000 lines) and is released both as source and as a precompiled Windows binary. It uses a modified bigdealx, called 2.0, but the dealing part is the same as the old 1.2 version. No player should be able to figure out from the hands alone that this set of software is used.

For the random session keys in this document, I use a cryptographically secure random function[i].

## At time T0

The first period starts. The tournament organizer (from this point on abbreviated to ORG) starts thinking about the tournament and begins preparation. I envision this to be at least two weeks in advance of the first session of play, but maybe even earlier.

He starts up SquareDeal and tells it a new tournament is coming. He needs to supply a short name, to be used as filename prefix. Let us be original in this document and call it *test*. The program will eventually make two files, *test.sqd*, which contains the description of the tournament and will be published before the tournament and *test.sqk* which will contain the keys and will be published after the tournament.

Squaredeal has two sets of functions, the first before publishing and the next after it. For a new tournament we start with the first.

There will be four options:

1. Set Tournament Name
2. Set Delayed Information Description
3. Add phase of tournament
4. Publish

The tournament name is just a string for description and will not be used for any operational purpose. Something like “Bermuda Bowl 2023 Oct 2, Vatican City”.

The delayed information description describes which specific information will be used as delayed information. More about this later. Something like “DJI index September 29”.

Each tournament is expected to have various phases, like Round Robin, Quarter Final, etc. Each phase will have multiple sessions.

The information for each phase contains the following:

1. The number of sessions
2. The number of boards per session. This can either be a simple number, like 16, or a range, like 17-32. It can also be a list of things, like 1-16,17-32. In the latter case the program will cycle through this list, in this case 1-16 and then 17-32 and then 1-16 again. This can be abbreviated to 2x16. Writing 17-32,1-16 also works, same cycling but starts with 17-32 (Sometimes the duplication room likes this)
   It is also allowed to give a ? as the number of boards. This signifies that you are not sure yet. When making hands for this phase you will be prompted for the number of boards.
3. The file prefix. This is identical to the normal one for BigDeal with one exception: any string of # characters will be replaced by at least the same number of digits for the session. So, suppose the prefix given is bbrr## and we generate session 7, the file prefix will be bbrr07. If you do not specify any # characters, the program will add a correct number of them at the end.
4. The description. A string, non-interpreted, that will be displayed when making the hands. Same trick with the # characters. Example: “Bermuda Bowl Round Robin #/23”. Again, if no # characters are specified the program will add something reasonable at the end.

The ORG can enter and exit the program various times and continue where he left off. If he made an error, and thinks he knows what he is doing, he could edit the SQD text file.

## At time T1

At a certain point the ORG is done preparing for the tournament. He chooses the *publish* option (#4) which terminates the first function of the program. Apart from writing all the info in the description file the software will generate a (very) big random number for each session[ii], and stores it in the .sqk file. A cryptographic secure hash will appear in the description file as a signature to be used for checking later.

Now the ORG receives instructions telling him to publish the .sqd file and keep the .sqk file *very very* secret.

Publishing the file can be done in various ways. As an example putting it on the tournament website and/or emailing it to all participants. We will see how this turns out. (The Polish Bridge Federation uses github to store it)

Keeping the key file *very very* secret would require at the least an encrypted USB stick or similar measure. Keep in mind that if the contents of this file leak before the end of the tournament players will be able to generate the hands themselves before the session.

At this point interested players can do something. They can make a copy of the .sqd file as published. There is not a lot they can do with the file now, but the file contains a signature of all session keys, and they need that after the tournament to check. If nobody makes a copy a devious ORG could change the keys and repost the .sqd file with a new signature.

Time T1 could be the same day as T0, or much later, but it should be some two weeks before T2.

## At time T2

This is the point where the Delayed Information comes to life. What is the Delayed Information (DI) and why is it needed? The DI is any info, chosen by the ORG, but such that everyone will be able to figure out that the ORG could not have known it at time T1. A DJI index, the gold price, the drawing of the Italian state lottery, anything is OK, including combinations of things. The only rules:

1. The ORG cannot have known the info at time T1. Preferably nobody should have been able to know it then.
2. At tournament time anybody should be able to know this information. Preferably it should stay known forever, for everyone to see.

Why do we need it? If the hands for the sessions would only depend on the session keys the ORG could plan them in advance. As we will see later the value of the DI modifies the session keys in a major way. If the value of the DI contains a reasonable number of bits (20 or so) it becomes almost impossible for the ORG to change statistical properties of hands. Even extremely suspicious players would be satisfied with 40 bits. Picking 6 from 49, say a Lotto type game produces about 24. Combine that with the DJI and you are in a very safe zone.

Anyhow, at time T2 the ORG calls the program again which will now perform the second set of functions and sets the now known DI, stored as DV in the description file. The players can look up the number now if they wish but can also do that later.

## At time T3

Tournament time is almost there. The duplication room is being set up and the ORG needs to generate hands. Call the program and use the *generate hands* option. You can specify a phase, and within the phase one or more sessions. Phase numbers can be a single number or * meaning all sessions of all phases. Session numbers can be a single number (make one session), a list like 1-10(make the first 10 sessions), or a * meaning all sessions. If you used the ? possibility for number of boards you will be prompted here for the number. That can be just something like “16”, but more complex things like “1-7,8-14,15-21” or “3x7” are also allowed here.

The ORG has no choice now, the hands will follow straight from two(three) pieces of information, the session key, the DV (and a possible reserve set option). This normally finishes what you can do. There is however one emergency option: if as an ORG you screw up, or someone from your staff does, it could be that in hopefully rare circumstances the hands for a session leak out (it has happened for example that hand records from session 4 were distributed after session 3). So, there must be a way to make reserve hands. This option exists; you can make a reserve set for each session.

However, having made reserve sets, the ORG is obliged to tell the players this. This can be done now and/or at time T4. An honest ORG that needs to do this more than once per five tournaments is very sloppy.

A reserve set is made using the session key, the DV and the word reserve, the planned sets use the word original. All of this is hidden from the ORG.

The keys sent to BigDeal are very large, larger than the current possibilities of BigDeal, which currently *only* has 320 bits available for the session key, which means it can generate $2^{320}$ sets of hands. This is about a million times a million times a million times the number of atoms in the known universe. It is enough.

When generating hands for the sessions there is an automatic option if the board numbers differ per round. For example, suppose a phase is 7 rounds with board numbers 1-10,11-20,21-30. Then each file of hands will contain 10 boards. For duplication purposes it could be handy to have files containing board numbers 1-30. If some sessions of a phase are made together (for example with the * option) and if the formats you generate are PBN (or some untested ones), they will be combined automatically. So, for example if sessions 1 to 3 are in files RR01.pbn, RR02.pbn and RR03.pbn there will automatically appear a file called RR01-03.pbn containing the first three sessions. The DUP format is too weird; I strongly recommend not making or touching a DUP-file ever.

## At time T4

The tournament is finished. The ORG now publishes both the updated .sqd file and the .sqk file. He also states if he has used reserve sets and which (and why). Sometimes he might have needed to improvise, and he can state how. Here his responsibility stops.

Interested players can now download these files and run the software themselves. The software will check the signature of the keys (or they can use standard SHA-256 software) and the players can now make all the hands of the tournament, comparing them to what they played. Any mishap here requires investigation and detailed explanation by the ORG.

Hopefully it will never come to that, and furthermore before starting to accuse ORG's it would be a good idea to check the software. Contact me at sater@xs4all.nl, maybe I screwed up.

## Connection to BigDeal

This software uses the bigdealx version of BigDeal. Normally the ReadMe file states that you should not do this. Given that I claim I know what I am doing I still use it.

There are three minor modifications:

1. BigDeal used a 160-bit hash of an owner string as extra entropy. Because now the ORG and the suspicious player must use the same starting value, I cannot use this owner string.
2. Normally using dangerous flags to BigDealx causes it to scream and convince you not to do it. We changed it not to scream when using the right flags.
3. Normally BigDeal in the x version complains very loudly of regenerating sets of hands. With this software regenerating sets of hands could happen easily if you just lost your session 3 of phase 2 and must remake it. Now we do not complain.

Implementing these three changes was done by adding a -W option (Wizard…) with a string as argument to replace the owner string. This string is the first half of the session key (about 175 bits for a 160-bit hash). The other half of the session key, and the (base64 encoded) DV and the original/reserve words are added with -e switches.

So, pointing you to the BigDeal online info, the hands in BigDealx with -W string are now generated by running the following three things through a RIPEMD-160 hash:

1. The 160-bit hash of the first half of the session key
2. The 160-bit hash of the second half, plus the rest
3. An increasing counter

After that all other normal BigDeal things happen.

## File formats

The formats of the .sq? files are:

# Test.sqd

### Before publish

```
# Description file of tournament for program squaredeal 2.0.5
#
TN Test Tournament
DI Vatican City August lottery
# Description of phases of tournament
# Per phase a line with SN nessions:nboards:filename:description
SN 17:1-10,11-20,21-30:round##:round robin #/17
SN 4:1-16,17-32:sf#:Semi Finals #/4
SN 6:20:fin#:Finals #/6
#
# Until published this file may be edited if so wished
```

### After publish

```
# Description file of tournament for program squaredeal 2.0.5
#
TN Test Tournament
DI Vatican City August lottery
# Description of phases of tournament
```

```
# Per phase a line with SN nessions:nboards:filename:description
SN 17:1-10,11-20,21-30:round##:round robin #/17
SN 4:1-16,17-32:sf#:Semi Finals #/4
SN 6:20:fin#:Finals #/6
KH 8d00eb6b5b0ae9f99791be42888a8d90005e3745ca206a1bd7690fce486f5132
```

After setting delayed info value

```
# Description file of tournament for program squaredeal 2.0.5
#
TN Test Tournament
DI Vatican City August lottery
DV 3 6 9 13 26 40
# Description of phases of tournament
# Per phase a line with SN nessions:nboards:filename:description
SN 17:1-10,11-20,21-30:round##:round robin #/17
SN 4:1-16,17-32:sf#:Semi Finals #/4
SN 6:20:fin#:Finals #/6
KH 8d00eb6b5b0ae9f99791be42888a8d90005e3745ca206a1bd7690fce486f5132
```

# Test.sqk

```
1,1:a3GyE3l6XLuooSmLnK0WcgU8RDRQYk1DJmumWzs2p6Np7h0Vjm0ya6djDI7E
1,2:04NmdiAygLMcdrbdUkSwTEZv4pypBmH1hkSNyzldQ1hQsfuHBJnnx2WUehM8
1,3:ZxjEeZubfevg3BBIRHyOJBjuXKFOCAydVUGEgQyKhUe0taoZehhPeJq2voY0
1,4:OJgsIPogAMAiZRPlPG783NAWpr7L410urOxLHE98SBGGvqBvk6cKaJsuzKE9
1,5:gICcDpKxUkm3fSc5ap8u0Wj0je8A5tq7mE9X5N6iC3bOwRXgOHZdbTenT1V6
1,6:KMbJCE2t8XV9AarQO5gbPooliGboXxBr1AGocvlm0WTBathZP5jcesOx1CrI
1,7:SLC9oniQC7bpa4lDIoPM00rnHVXFnBZFtZIXgO0LI7Pv2ziiGvFaw8tIzYKz
1,8:zvzNEFkdaUZy439VPCqR1Vzz5YdEkxwh9i452OedKACiS487SnwIfnNnC444
1,9:fd9HcbasMS7TuDx7ZZ7IepLwt9j0s2JbvQrQhOvVtGjJyEuL1yQHe4NiD75C
1,10:ZKJjAMv7WF4gTFfAD2YCDxh40y4INlpy9gcC2XdLdV78YBbtz6q2mpVQfXMF
1,11:MP0QWq0DVLqldvzqJCthlEdjuyiurvjcJ7HN2G9b7CPIMWUWJ1yjx7qvljon
1,12:2UrUwZWWHMitvXOqRVfyCGz8KnkYoL7DbRj4d0LvdgCF9rmG4PEf1492zcxI
1,13:4oiJ9hJZ9ZLJIzlVrJWgtcqy4XyjQhhE2uDxm0GFc8VlWFtcLqIvg2ZGaK0y
1,14:UEow5P0QbRKL6ZZ5RbT9kqYsT5XyXUql4r6VtuDxCjzhnVgExRxU3AYtVS8R
1,15:lJrSYhi2XdV96GaDOQxUW4nwiJP4jDeSuQsuvHcLLxaEJzIIH9Xjzys1xyMw
1,16:sLt9S52YdVwYeinF0Yw63MmOrMRnrDhwdYM0jDZBeSNaC7LiMIdxY3LfOuNH
1,17:YfIk5m6UtVNwuGsMUbpLcruWOkOAAez2hZYIHiiwfPKbfWfrCjPRwvt9wLfA
2,1:5hVAj8aILG4gzwVi6u280qbbaPHyKlp6xLW3ulJbSf0xKXBvY0aAc674iHEg
2,2:mwrbE39sj3iajQxXL8llUY2aomoMuFRSSCyHLakRf7FaXqK5yGXaaHJU5v2Z
2,3:zmdvszEikiRg1PuVp8mpJcXwW7w41Cv8gK5Ddg0fLhr020Df4wqUtxDYkJPb
2,4:bTJJ9PYSb2bsipd64hPCkRMmU2wncjDGMORZl0njO9IywSC5WiaFPuCI7Yem
3,1:BozrEX4x3CaXxKQwQjmgqgecoWgGQfApeW8OPiWfXLoAw4ikEyEWce8RpJuE
3,2:9TIJybO2xC7fHy9wHuBMwyFtCfxTW4gNANHLl3TttDMWTbvLXgCZBdjbo993
3,3:g3D4qaF8DXNBwvHAY33H7N9wPjGk6DYlF57sJgeAM4D7FKKDVpCzsqsE8vPQ
3,4:y82JJcj5gT282kZf6TUyVJTC6s2fYwfz1IcUf4Pm5BWp17ljOohoTTmGLzrs
```

```
3,5:i2brjxIU07w6huFcWMK1obBgR4hvy5e4k4mkkdMJQZLhUWR4FF6kzDruBOoP
3,6:Ms3uhj0voy9LBvpDkjrESBfFJO85rmJy30ylaqzPW5LZU69YhlXnpslSHhSU
```

For the key hash I use the SHA-256 function. The annoying detail you must figure out is the line termination. On Unix type systems line termination is \n or NL, while on DOS like systems it is \r\n or CRLF. Normally you could not care less, but for this key hash it is important you do it consistently. I chose to use the DOS like convention.

## Conclusion

This software is handy to use for larger tournaments, even if you are not interested in the *proving innocent* stuff. Just the generation of many sets of hands with consistent naming and joining of sets for duplication purposes is worth it.

---

[i] perl Bytes::Random::Secure::random_string_from function which is supposed to use the OS secure random generator.

[ii] 60 characters with 62 possibilities equals about 357 bits.