0.0.1 slide1

Risk is a popular strategy game in which competing players aim to control all regions on a board by moving pieces and attempting to overthrow neighbouring regions.

Each region within a player's control has some number of pieces placed onto it. To attack a neighbouring region, a player gambles some of their pieces against the opposing player's pieces.

Risk was originally a board game, but has since been turned into an online game as well.

0.0.2 slide2

In the online variant, there is a mode known as "fog of war", where players can only see the number of pieces that are placed on neighbouring regions. Therefore, this is only played online and in a trusted setup, with players communicating indirectly via a server.

0.0.3 slide3

My proposition is to play fog-of-war risk in an untrusted setup, for example in a peer-to-peer network. The same guarantees should be made as in the trusted setup, but without the mediating party (which is the server).

0.0.4 slide4

Besides cryptography and decentralised networks being of personal interest, there are further benefits to the expansion of cryptographic methods and their applications to decentralised platforms.

decentralised platforms offer many benefits to their users, such as longer lifespans due to being community-ran. A good example of this is networks such as usenets and IRC outlasting many social media platforms, as they still have a community backing them.

Additionally, decentralised platforms are more resistant to censorship, as they are run by their users, and can be run from any jurisdiction, to avoid laws that may restrict the operation of platforms in certain countries. This also helps promote anonymity and privacy, as the platform isn't being run by a company that may have a legal obligation to collect certain information, or may just collect information for the sake of it.

Finally, the main benefit is that decentralised platforms promote user freedoms, as the code can be modified and audited to tailor it to the user's needs.

However, decentralised platforms are exposed to a unique set of challenges than centralised platforms. Secure storage of information is difficult, and attacks against decentralised networks can still be devastating.

For example, Tor experienced unique attacks against its infrastructure, including denial-of-service against old tor v2 addresses, causing the network to go offline for many users.

To mitigate security risks, platforms such as torrents revert to centralisation in the form of trackers to validate file chunks being sent via peers. The idea is that we want to avoid this centralisation as much as possible.

0.0.5 slide5

To achieve this, I will be using many of the standard cryptographic protocols such as AES and RSA, but along with the Paillier cryptosystem and zero-knowledge proofs, which are some less appreciated areas of cryptography.

Paillier is of interest as it has an interesting additive homomorphic property, where manipulating cyphertexts manipulates the underlying plaintexts in predictable ways.

Zero-knowledge proofs are currently in use in the blockchain ledgers monero and zcash. They use bulletproofs and zksnarks respectively to obfuscate transaction amounts, recipients and senders. I will be using a scheme described by Damgard et al, which allows construction of ZKPs for a cyphertext being non-zero

On a more practical note, my implementation is targetting the browser. This is of interest purely as it applies some particularly new APIs such as weblocks and bigint.

0.0.6 slide6

My network is an emulated P2P environment using websockets. I went for an emulated environment as it relieves me of implementing UDP hole-punching, and furthermore websockets are a simple way to transfer data in a simple format between clients.

0.0.7 slide7

Risk relies on dice-rolling mechanics. To achieve this, I have implemented a scheme to produce shared random values without a beacon by using commitment schemes. Each player submits some encrypted noise, and then each player submits a decryption key to yield a random value within a known range.

0.0.8 slide8

As part of the Paillier cryptosystem, the generation of large primes is required. I implemented Rabin-Miller with the ECMAScript 2019 BigInt standard to produce primes of 2048 bit length, for a combined key size of 4096 bits.

0.0.9 slide9

Furthermore, I implemented the paillier cryptosystem. This came with some difficulty, as the bigint spec is not followed correctly by major browsers.

0.0.10 slide10

Finally, I have a P2P implementation of standard risk. The map is reduced to make testing quicker.

0.0.11 slide11

The next steps for the implementation are to blend the paillier cryptosystem with the game itself to get the fog-of-war variant. This requires the implementation of a zero-knowledge proof scheme that is described by INSERT REFERENCE.

I also want to improve the implementation somewhat to reduce other more general attack surfaces. For example, preventing players from stopping play by not responding to messages, which is currently an effective strategy for a losing player. This also means gracefully handling disconnects.

Further analysis is also necessary of the optimisations I made to the paillier cryptosystem, as certain computations were subtly changed to ensure that the size of intermediary values didn't exceed the upper limit on big integers. This may have introduced a modular bias, which would be breaking.

Following this, I plan to do a more general analysis of the system to check its security and benchmark.