Superposition Attacks on Cryptographic Protocols

Ivan Damgård, Jakob Funder, Jesper Buus Nielsen

Aarhus University

Louis Salvail, Université de Montréal

Usual model of attacks on classical cryptographic protocols

Adversary

query

response

……

"Oracle" - Representing players in the protocol under attack

Examples.

- Attacking a ZK protocol: oracle = the honest prover. Query: the verifier's challenge. Want to show that Adversary learns nothing about the prover's secret.

- Attacking a secret sharing scheme. Query= set of players the adversary wants to corrupt. Response= shares held by corrupted players. Want to show that adversary learns no information on secret.

What if the adversary is quantum?

In previous work: same model, only the adversary is now a quantum machine.

Several known results that establish security in this scenario (ex: [Watrous: zero-knowledge for quantum verifiers]).

Question: why is the communication between adversary and oracle still classical?

Answer: because honest players are classical, so can assume that measurements are (implicitly) done on anything received.

However…

What if honest players are also quantum?

Could happen, even if protocol is supposed to be classical:

- Honest player apply quantum computing locally to gain efficiency
- Classical MPC used as subrutine in quantum MPC [Crépeau et al].

Now seems less obvious that a quantum adversary must communicate only classically with honest players.

Example:

A prover in a ZK protocol implemented as a small quantum component sitting inside a mobile device. If adversary gets hold of the device, who knows what could happen?

Superposition attacks on classical cryptographic protocols

Adversary

Query=

$\Sigma_{q\varepsilon Q,x}\ \alpha_{x,q}|q>|x>$

"Oracle" - Representing players in the protocol under attack

Response=

$\Sigma_{q\varepsilon Q,x}\ \alpha_{x,q}|q>|x+R(q)>$

q: classical query
Q: set of allowed queries
R(q): the response to query q in classical game
NOTE – we have simplified the model:
1)in most cases, Res() will be probabilistic, so the response will actually be a mixed state.
2)We ignore for now the question of how the inputs are chosen.

Superposition attacks on classical cryptographic protocols

Adversary

Query=

$\Sigma_{q\varepsilon Q,x}\ \alpha_{x,q}|q>|x>$

"Oracle" - Representing players in the protocol under attack

Response=

$\Sigma_{q\varepsilon Q,x}\ \alpha_{x,q}|q>|x+R(q)>$

The basic question:

Assume the protocol is classically secure for a certain of queries Q'.
Which set Q can we allow in the quantum query and still be secure?

An example to illustrate the problem

Secret-sharing among 2 players. Secret s is a bit, value in F_2

$s = s0 + s1$.

Player i holds si, i=0,1.

In classical game: Adversary can ask for shares belonging to some subset of players. Secure if subset has size 1.

➔ Set of allowed queries is Q'= {0, 1}

In the quantum game, allowing queries in Q' is insecure! Send query
$(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$
Response will (essentially) be

$(|0\rangle + (-1)^{s0+s1} |1\rangle) (|0\rangle-|1\rangle)$
Now do a Hardamard transform on first register

$|s0+s1\rangle (|0\rangle-|1\rangle)$
Measurement gives you the secret with prob. 1 (same idea as Deutch-Josza).

All is not lost

Can characterize the cases where perfect quantum security for secret-sharing is possible.

Let Q' be *adversary structure* for classical secret sharing scheme.

Q' = family of subsets of players that adversary may ask for in classical game, and still he gets no info on secret.

Theorem. We have perfect security in quantum game if and only if: Consider any two subsets A1, A2 in the adversary's superposition query. Then the union of A1 and A2 is in Q'.

I.e. the family Q of subsets allowed in quantum query consists of all "small enough" subsets.

In threshold case: if you are classically secure against corruption of t players, then you have perfect quantum security for corruption of t/2 players.

Sketch of proof

Adversary's query

$$\sum_{A \varepsilon Q,x} \alpha_{x,A}|A\rangle|x\rangle$$

For a fixed secret s and choice r of randomness used in the secret sharing scheme, response will be

$$|Res(s,r)\rangle = \sum_{A \varepsilon Q,x} \alpha_{x,A}|A\rangle|x+shares(A,s,r)\rangle$$

Where shares(A,s,r) = set of shares given to players in A, for secret s and randomness r. State actually seen by adversary for secret s is a mixture:

$$\sum_r Pr(r)|Res(s,r)\rangle\langle Res(s,r)| =$$

$$\sum_{A,A' \varepsilon Q,x,x'} \alpha_{x,A}\alpha^*_{x',A'}|A\rangle\langle A'|$$

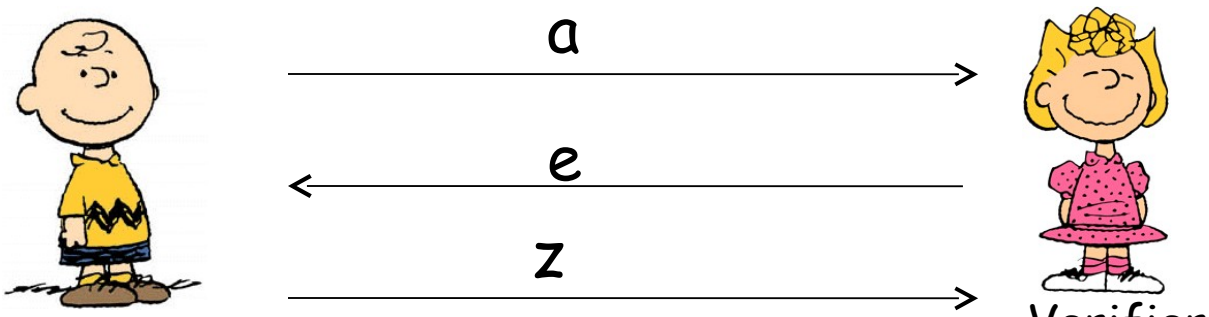
$$\sum_r Pr(r)|x+shares(A,s,r)\rangle\langle x'+ shares(A',s,r)|$$

Independent of s, if and only if we have classical security against corruption of A and A' simultaneously.

Zero-Knowledge Protocols

Consider a protocol in the standard 3-move form (Σ-protocol),
Verifier sends random challenge e as second message.

Common Reference String: pk, $E_{pk}(0)$



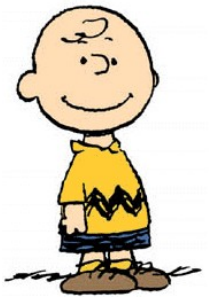
Prover,
Proves NP-statement x is true
or CRS contains 1

Verifier

Is only honest verifier ZK, but can get ZK for general verifiers in various ways. For instance in CRS model..

Zero-Knowledge Protocols

Common Reference String: pk, $E_{pk}(0)$



a

Superposition of e-values

Superposition of z-values

Verifier

Prover,
Proves NP-statement x is true
or CRS contains 1

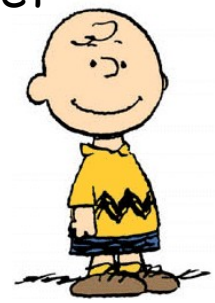
What if verifier is corrupt and quantum?

We do not know what happens in general, but using specific construction, can get ZK for all of NP and soundness even if prover is quantum..

Basic idea to get ZK

Common Reference String: pk, $E_{pk}(0)$, PK: public key for commitment scheme

Prover secret-shares his witness



Prover, Proves NP-statement x is true or CRS contains 1

a= commitments to shares of witness

e= ask for subset of t/2 shares

z= open relevant commits

Verifier

Intuition: Assume secret sharing scheme has t-privacy and commitments are unconditionally hiding: OK to open t/2 shares, even if Prover is forced to answer several e-values in superposition.

But how do we know that the committed shares really determine a correct witness??

How to check that shares are correct

Use "MPC in the head" [IKOS].

P emulates in his head the execution of an unconditionally secure MPC protocol π for n players where witness is shared among the players initially. Protocol checks that witness is correct, i.e., it is a witness for x, or a witness that CRS contains 1. All players output yes or no.

Need: π is secure against active corruption of t players.

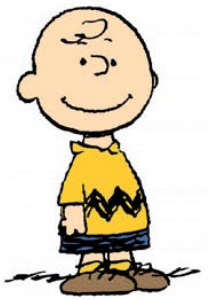
Can interpret emulation of π as a secret-sharing scheme, where
Share no. i = View of player i

When verifier sees a number of such shares, can check that all "opened players" output yes and views are consistent. [IKOS] shows that if correct witness does not exist, verifier accepts with negligible probability.

Manipulating the reference string to get soundness and ZK

Common Reference String: pk, $E_{pk}(0)$,

PK: public key for unconditionally binding commitment scheme



a= commitments to shares of witness

e= ask for subset of t/2 shares

z= open relevant commits

Verifier

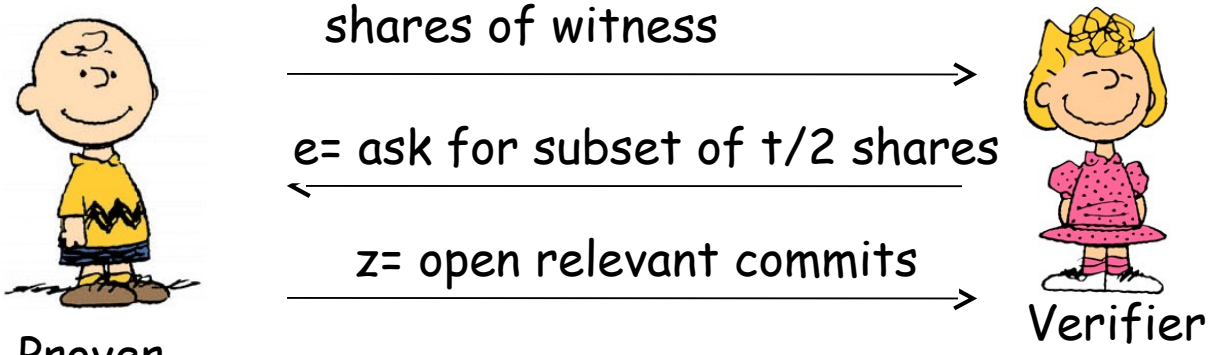
Prover, Proves NP-statement x is true or CRS contains 1

Soundness: commitments unconditionally binding, so soundness proof of IKOS applies even if prover is quantum. Since CRS contains 0, Prover must use witness for x to survive.

Manipulating the reference string to get soundness and ZK

Common Reference String: pk, $E_{pk}(1)$,

PK: public key for unconditionally hiding commitment scheme



ZK: simulator will put encryption of 1 in reference string and simulate by following the protocol. PK will be key for unconditionally hiding commitment scheme. Works, assuming key types are indistinguishable and encryption is secure, even against a quantum verifier

Instantiating commitments and encryption scheme

Can use Regev's LWE-based scheme for both.

- Scheme believed to be semantically secure against quantum adversary, so can use a Regev-key as pk.

- Proof of security for Regev's scheme works by showing
  - if public key is chosen randomly and independently of public key,

    ciphertexts statistically hides message
  - distinguishing real and random public keys reduces to LWE.

Means we can set PK to be random or real Regev-key and commit by encrypting under PK.

General Multiparty Computation

n players P1,..,Pn, have inputs x1,…,xn, want to compute (y1,…,yn)= f(x1,…,xn) securely, even if t players are corrupted by an adversary.

Here, consider only static, passive corruption:
- set of corrupt players determined before protocol starts.
- adversary just observes views of corrupt players, everybody follows protocol.

In this case, classical security means:
- result is correct
- corruption of set A reveals only (xi, yi) for Pi in A.

Privacy proved by simulation: an efficient simulator S must exist. S gets corrupted set A and (xi,yi) for Pi in A as input and must output the view of players in A, with distribution as in real protocol.

A Model for Attacks on MPC in Superposition

Have defined a UC-like model for superposition attacks where

- Adversary chooses inputs for all players, possibly in superposition.
- Makes a query containing (superposition of) set(s) to corrupt
- Gets back the views of corrupted players executing the protocol.

$$\sum_{A \varepsilon Q,x} \alpha_{x,y,A} |A\rangle_{set} |x\rangle_{IO} |y\rangle_{view}$$

$$\sum_{A \varepsilon Q,x} \alpha_{x,y,A} |A\rangle_{set} |x + IO(A)\rangle_{IO} |y + view(A)\rangle_{view}$$

What a quantum simulator S must do

One possible model: adversary's query is sent to an oracle (ideal functionality) that supplies input/outputs for corrupted players, then S gets the result, and must simulate the views of corrupted players.

$\Sigma_{A\varepsilon Q,x}\ \alpha_{x,y,A}|A\rangle_{set}|x\rangle_{IO}|y\rangle_{view}$ Goes to Ideal Func.

$\Sigma_{A\varepsilon Q,x}\ \alpha_{x,y,A}|A\rangle_{set}|x+IO(A)\rangle_{IO}|y\rangle_{view}$ Is given to S

$\Sigma_{A\varepsilon Q,x}\ \alpha_{x,y,A}|A\rangle_{set}|x+IO(A)\rangle_{IO}|y + view(A)\rangle_{view}$ Output from S

Note: since views are probabilistic, output from S as well as from protocol are mixed states.

Results on MPC

$$\Sigma_{A \varepsilon Q,x}\ \alpha_{x,y,A}|A\rangle_{set}|x\rangle_{IO}|y\rangle_{view}$$

If adversary is allowed to choose any state for the IO register, simulation is impossible:
adversary can put the register in uniform superposition over all x values, then adding in IO(A) does not change the state, so S gets no information on IO(A).

Letting S process the query before it goes to the ideal functionality does not seem to help.

But if we require x=0, some positive results..
Can simulate protocols with 2 different inputs, such as secret sharing a bit where also dealer may be corrupted. We believe it can be done in general, but this is still open.