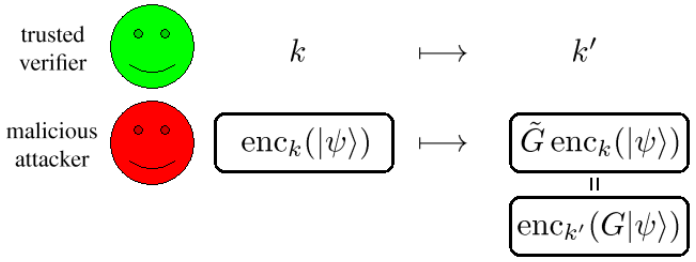
Lecture 4: Quantum computing on authenticated data

In this lecture we introduce the notion of *quantum computing on authenticated data (QCAD)* and show that the trap scheme admits QCAD. We then list some applications of QCAD.

1 What is QCAD?

It is helpful to think of two parties: a trusted *verifier* who prepares authenticated data with secret classical key k and a malicious *attacker* who is to act upon the authenticated data without knowledge of k. The goal is to construct an authentication scheme with the property that for each gate G belonging to some universal set of gates there exists a *gadget* circuit $\tilde{G}$ that the attacker can perform on authenticated data so as to implement a logical G. Furthermore, we require that the gadget $\tilde{G}$ be *independent of the choice of classical key* k so that it may be implemented by an attacker without knowledge of k.



Normally, any non-identity gadget $\tilde{G}$ would invalidate the authenticated state. We therefore require a scheme which allows the verifier to validate the state again simply by updating the classical key $k \mapsto k'$. Moreover, by updating the key in this way the verifier effectively *forces* the attacker to apply the desired gadget $\tilde{G}$ as otherwise the state would fail verification under the updated key k'.

2 A universal gate set for the trap scheme

In order to achieve QCAD for the trap scheme based on code C, we require that C be a *self-dual CSS code*. The seven-qubit Steane code is one example of a self-dual CSS code that suffices for this purpose. Specifically, it follows from Proposition 4 of Lecture 3 that if our trap scheme is to have security $(2/3)^{d/2}$ then it suffices to base the trap scheme upon the Steane code nested a sufficient number of levels so as to achieve distance d.

Our universal gate set consists of the following gates, listed in the order they are presented in the following subsections.

1. The standard single-qubit Pauli gates, denoted X, Y, Z.
2. The standard two-qubit controlled-NOT gate, denoted CNOT.
3. The single-qubit i-shift phase gate, denoted K and specified by $K : |a\rangle \mapsto i^a|a\rangle$ for $a \in \{0, 1\}$.
4. The single-qubit $\pi/8$-phase gate, denoted T and specified by $T : |a\rangle \mapsto e^{ai\pi/4}|a\rangle$ for $a \in \{0, 1\}$.
5. The standard single-qubit Hadamard gate, denoted H.

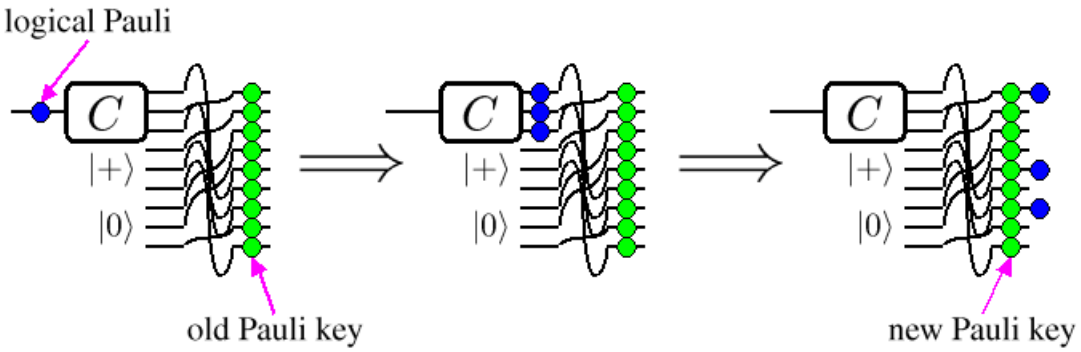
This gate set is redundant in the sense that only $\{\text{CNOT}, H, T\}$ are required to achieve universality. Indeed, $T^2 = K$ and $K^2 = Z$, so why bother listing these extra gates? The answer is that the gadgets for the Pauli and controlled-NOT gates are very simple. By contrast, the gadget for K is a magic state gadget that requires CNOT and Y and the gadget for T is a magic state gadget that requires CNOT, X, and K. Thus, in order to get a T gate we must "bootstrap" our way up from Pauli gates, CNOT, and K. (If an application calls for only Clifford circuits on authenticated data then the T gate construction can be ignored, as $\{\text{CNOT}, H, K\}$ suffice to generate the Clifford circuits.)

Some gadgets are simple whereas others are more complicated. Gadget design is borrows ideas from fault-tolerant quantum computation.

2.1 Pauli gates

The gadgets for each Pauli gate X, Y, Z are empty. In order to implement a logical Pauli gate in the trap scheme the attacker does absolutely nothing to the authenticated register and the verifier simply updates the Pauli key according to the desired Pauli gate.

In particular, logical Paulis for the seven-qubit Steane code admit straightforward bitwise implementations. Thus, the verifier can implement a logical Pauli Q in the trap scheme by modifying the Pauli key according to $P \mapsto PQ^{\otimes 7}$ where $Q^{\otimes 7}$ is applied to encoded data, leaving the Pauli key for the trap registers unchanged.



2.2 The controlled-NOT gate

The gadget for a controlled-NOT from logical qubit a to logical qubit b is a straightforward bitwise CNOT applied from each physical qubit of a to its corresponding physical qubit in b.

To see that this simple bitwise gadget implements logical CNOT recall that every CSS code (including the Steane code) admits a bitwise implementation of logical CNOT. Moreover, the CNOT applied bitwise to the trap registers acts trivially on those registers:

$$\begin{aligned} \text{CNOT} &: |0\rangle|0\rangle \mapsto |0\rangle|0\rangle \\ &: |+\rangle|+\rangle \mapsto |+\rangle|+\rangle \end{aligned}$$

Finally, observe that bitwise CNOT is invariant under permutation of the physical qubits *provided that both data blocks are subjected to the same permutation*. Here we employ the fact that encode-encrypt schemes allow for the re-use of code keys as discussed in Section 1.4 of Lecture 3.

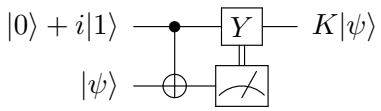
The Pauli key is updated according to the well-known effect of CNOT on Pauli operations. In particular, if the Pauli keys for the ith physical qubit from both data blocks are $X^p Z^q, X^r Z^s$, respectively, then the updated Pauli keys for these physical qubits are $X^p Z^{q+s}, X^{p+r} Z^s$.

[*** picture ***]

2.3 The i-shift gate

Readers familiar with the Steane code know that a bitwise K gate applied to each physical qubit implements K^* on logical data. At first glance one might therefore hope that the K gate, like CNOT, admits a simple bitwise gadget under the trap scheme. Unfortunately, trap codes do not admit bitwise implementation of the K gate even if the underlying code does admit such an implementation. Bitwise implementation fails for trap codes because the trap qubits prepared in state $|+\rangle$ are mapped by K to $K|+\rangle = |0\rangle + i|1\rangle$. A trap qubit in this state is detected as a Z-error with probability $1/2$.

Instead we require a more complicated magic state gadget for K that uses only Pauli and CNOT gates together with measurement in the computational basis. Our gadget is a simple modification of the well-known fault-tolerant construction for the $\pi/8$-gate. The logical gadget for the K gate is depicted as follows.



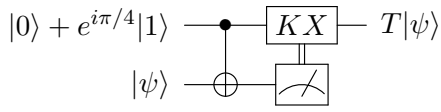
Here $|\psi\rangle$ denotes the arbitrary state of the data qubit; the magic state for this gadget is $|0\rangle + i|1\rangle$. The physical gadget to be implemented by the attacker on authenticated data is derived from the above logical gadget by replacing the input qubits with authenticated registers and by replacing the logical CNOT, Y, and measurement with their respective physical gadgets.

Once the authenticated magic state has been prepared, all the gates in this gadget except the measurement can be implemented solely by the attacker. The verifier's knowledge of the secret key is required in order to decode the measurement result, which indicates whether a Y correction is needed.

Since Y is a Pauli gate and since Pauli gates require no action from the attacker, this gadget can be implemented with only one-way classical interaction from attacker to verifier. In particular, the attacker implements the measurement by bitwise measurement of physical data as described in Section 1.3 of Lecture 3. The verifier decodes the measurement result (and checks for tampering in the process) and then implements the Y correction (if it is needed) simply by updating the Pauli key for that qubit as described in Section 2.1.

2.4 The $\pi/8$-phase gate

The gadget for the T gate is a magic state gadget that is very similar to the magic state gadget for the K gate described Section 2.3 and consequently much of the discussion from that section applies here. The well-known fault-tolerant construction for this gate can be used verbatim as the logical gadget for the T gate in the trap scheme. That construction is reproduced below.



The magic state for this gadget is $|0\rangle + e^{i\pi/4}|1\rangle$.

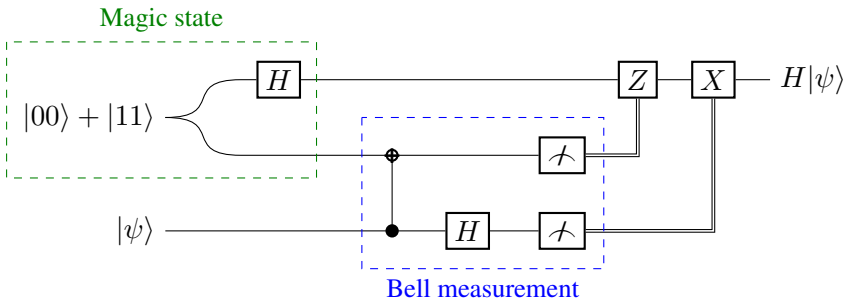
Whereas the gadget for K presented in Section 2.3 specified only a Pauli Y correction, the correction required in the T gadget is the Clifford gate KX. Two-way communication between verifier and attacker is required to implement this gadget because the verifier must inform the attacker as to whether to apply a K gate. Naturally, this K gate, if it is required, is achieved via the magic state gadget presented Section 2.3, which requires a separate magic state of its own.

Since K is not a Pauli gate subsequent computation on authenticated quantum data cannot proceed until the correction is applied (if it is needed). Thus, the verifier must decode the classical measurement result and reply immediately to the attacker with instructions as to whether a correction is required.

2.5 The Hadamard gate

Similar to the K gate, a logical H gate can be implemented under the Steane code by applying H bitwise to each physical qubit. One might therefore hope that the H gate admits a simple bitwise gadget under the trap scheme. Unfortunately, as with the K gate, trap codes do not admit bitwise implementation of H even if the underlying code does admit such an implementation. Bitwise implementation fails for trap codes because the $|0\rangle$ and $|+\rangle$ trap qubits are swapped by the action of bitwise H. Each trap qubit is thus in a state that is detected as an error with probability $1/2$.

As with the implementations of the K and T gates described previously we shall implement the Hadamard by a magic state gadget. Whereas the gadgets for K, T are compact and efficient, the simplest known magic state gadget for the Hadamard gate is the teleport-through-Hadamard circuit, which is a special case of the gate teleportation protocol of Gottesman and Chuang [GC99]. This circuit is depicted below. The magic state for this gadget is the two-qubit maximally entangled state $|00\rangle + |01\rangle + |10\rangle - |11\rangle$.



Implementation of the Bell measurement appearing in the above circuit requires that a Hadamard gate be applied to one of the two qubits immediately prior to measurement. At first glance this requirement might appear circular, as we require a Hadamard gate in order to implement the Hadamard gate. We claim, however, that it is possible to implement the Hadamard gate bitwise on authenticated data *provided that the qubit is measured immediately afterward* as is the case for a Bell measurement.

This claim is not difficult to justify. As mentioned above, the effect of the bitwise H gate is to swap the $|0\rangle$ traps with the $|+\rangle$ traps immediately prior to measurement. But it is trivial to modify any measure-then-decode procedure (such as that mentioned in Section 1.3 of Lecture 3) so as to take this swap into account.

As with the gadget for K presented in Section 2.3, the correction in our gadget for H is a Pauli gate. Hence, implementation of this gadget requires only one-way classical communication from attacker to verifier.

2.6 Misc remarks

1. The implementation of CNOT necessitates that each authenticated qubit share the same code key.
2. Clifford circuits can be implemented offline (no interaction with the verifier).
3. Any circuit can be implemented with only classical communication between attacker and verifier.

3 Applications

QCAD appears to be a powerful technique in quantum cryptography. Yet it has thus far found only limited application, despite being discovered over seven years ago. Here are some of the known uses of QCAD.

3.1 Multi-party quantum computation

QCAD was first discovered in 2006 when it was used in a secure protocol for multi-party quantum computation [BOCG$^+$06]. In that paper QCAD is built on a qudit authentication scheme called the *signed polynomial scheme*. Even with QCAD established, the protocol for MPQC is very complicated. Essentially, the protocol employs secure multi-party *classical* computation in order to assume the existence of a trusted classical third party who maintains authentication keys, decodes measurement results, and exchanges only classical information with the other parties. The parties in the protocol apply an arbitrary quantum circuit by shuttling authenticated quantum data around via teleportation (sending measurement results to the trusted third party) and employing QCAD.

3.2 Delegated quantum computation

QCAD was used to implement a form of *delegated quantum computation* [ABOE10], which allows for a *user* with extremely limited quantum computational power to "delegate" an arbitrary quantum computation to a more powerful *server* in such a way that the server learns nothing about what was computed and the user can verify that the server did not tamper with the computation. Their application required only a constant security guarantee and was applied only to circuits with an all-$|0\rangle$ input state and a single classical bit as output. As such, the user in this application requires only the ability to prepare constant-sized quantum registers in the first round of the protocol, thereafter exchanging only classical messages with the server.

3.3 Quantum one-time programs

A *one-time program* allows a user to evaluate a circuit exactly once on an input of his choosing, then self-destructs. One-time programs cannot be implemented without assumptions about secure hardware, but one-time programs for classical circuits can be achieved assuming only a hypothetical *one-time memory* device. QCAD was employed to show that one-time programs for *quantum* circuits can be achieved using only the same one-time memory devices used in classical one-time programs [BGS12]. The idea is to employ a classical one-time program to manage the key and handle all the classical interaction with the quantum user, who applies the quantum circuit to authenticated data.

References

[ABOE10] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. In *Proceedings of Innovations in Computer Science*, pages 453–469, 2010. arXiv:0810.5375v2 [quant-ph]. 5

[BGS12] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. To appear in CRYPTO 2013. arXiv:1211.1080 [quant-ph], 2012. 5

[BOCG$^+$06] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *Proceedings*

of the 47th IEEE Symposium on Foundations of Computer Science (FOCS 2006), pages 249–260, 2006. arXiv:0801.1544 [quant-ph]. 5

[GC99] Daniel Gottesman and Issac Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402:390–393, 1999. arXiv:quant-ph/9908010v1. 4