Tracking cryptographic keys and encrypted data using position verification

Máté Galambos\textsuperscript{1,2}, László Bacsárdi\textsuperscript{1}
\textsuperscript{1}Networked Systems and Services, Budapest University of Technology and Economics, Magyar tudósok krt., 2., H-1117, Budapest, Hungary
\textsuperscript{2}E-mail: galambos@mcl.hu

Abstract: Position verification is an emerging field of quantum cryptography. Its goal is to verify whether a distant communicating party is telling the truth about where they are. However, the problem is usually formulated in a way that the position is the only credential of that party, which cannot guarantee uniqueness. In this study, the authors show how a practically secure position verification algorithm – assuming it exists – might be used to track (i.e. repeatedly verify the position) of some unique key or cipher text. To achieve this, they rely on pre-prepared position verification data called trackers. They also propose three algorithms that implement their general tracking scheme and examine some questions related to their security. These implementations include shuffling trackers into valuable data and hiding their memory address through a random permutation; using CNOT operations to entangle valuable data and trackers; and using random qubit strings from which either trackers or secret keys can be produced at will. These methods may be used to track a diplomatic package or reveal the location of a malicious party during a denial of service attack.

1 Introduction

Position-based cryptography is an emerging subfield of quantum cryptography with no classical equivalent \([1]\). Protocols in this category are based on some form of position verification – a method whose goal is to confirm that a distant communicating party is telling the truth about where they are \([2–4]\).

If they tell the truth then they pass the verification (this was shown to be possible even in the presence of a limited amount of noise \([5]\), and if they lie about their position, they will fail the verification round with high degree of probability. (This probability is comparable to guessing an unknown key correctly.)

The procedure was originally designed to use the location as the only credential of this distant communicating party – which may be useful for some applications but limits other use cases. These other use cases include tracking something by repeatedly verifying its changing geographical location which would require identifying what we are tracking.

For example, tracking a diplomatic package or nuclear briefcase could be useful. We could establish geographical areas where they cannot be transported safely and verify that they did not enter these areas. If they get lost, actions could be planned based on their last known and verified location. Or if a nuclear briefcase is activated, its position can be verified, which could reveal irregular behaviour.

There were attempts at proving uniqueness of the communicating party. Some authors introduced protocols where a tagging device can be uniquely identified as long as malicious parties do not have access to a hidden bit string housed inside the device \([6]\). Others suggested using this unique tagging device to verify the geographical position of self-driving cars \([7]\). This paper highlights the practical need for verified tracking, but the results need to be handled carefully. The assumption that Attacker cannot examine the internal components of a tagging device is not necessarily realistic, and even if it would be, there is no guarantee that this tagging device is attached to a car.

Another similar use case is position-based authentication \([8]\). This method binds position verification to a given message and ensures that the message was sent from the alleged position. However, the goal here is different: the protocol should work with any message that the distant communicating party chooses and authentication verifies the origin of this message a single time. Therefore, this approach is not suitable for tracking.

To have a tracking protocol, two problems must be solved. (i) The protocol should remain secure even if whoever is carrying it is malicious. This includes verifiably protecting form copying attempts. If data can be copied then its location becomes meaningless. (ii) Repeated verification must be possible – the protocol should preserve at least part of the data. If the data is completely used up, and becomes unreadable or insecure after a single round of verification, tracking is not possible.

In this paper, we examine how these problems could be solved and how data could be tracked. Our method could be used to track the movements of a nuclear briefcase or a diplomatic package. It could also hinder denial of service attacks by verifying where packages were lost in a quantum network and therefore revealing at least some of the nodes which came under the control of a malicious party.

We give a general scheme of how this can be achieved as well as discuss multiple possible realisations of such a quantum-tracking scheme. We also examine some questions regarding security: we examine the probability that compromised integrity remains undetected in case of the shuffling protocol, and whether revealing information about the order of memory slots weakens the security in CNOT protocol.

Furthermore, we introduce a new implementation: the tracker keys. Finally, we examine how trackers could be produced using a hypothetical, practically secure position verification method if the number of Verifiers can be increased.

2 Position-based cryptography

To confirm the location of a distant communicating party (called a Prover), the cooperation of several reliable parties (called Verifiers) is necessary. The verification process is based on the impossibility of faster than light signalling, the No Cloning Theorem \([9]\) and – according to our current understanding – a limit on efficient instantaneous non-local quantum computation \([2, 3, 10, 11]\).

Whether position-based cryptography is secure or not is still an open question \([4, 11]\). There are known general quantum attacks that are inefficient but universal \([3, 12]\) – meaning they can be used against all quantum position verification protocols, yet they are extremely resource intensive which could render them practically infeasible. On the other hand, we do not know if there is a position verification protocol for which this general attack is optimal.
2.1 Position-verification process

During position verification, multiple Verifiers (with well-known locations) transmit shares of a question to the Prover. They time their messages so that these question-fragments reach the alleged position of the Prover at the same time, travelling at the speed of light.

On a side note: although there are variants of these protocols that use only one Verifier and multiple Provers [13], here we rely on only one Prover and later assume that the number of Verifiers and therefore the number of question shares could be increased beyond what is strictly necessary by at least one.

The Prover restores the question from the shares, calculates the answer (preferably in negligible time) and replies by transmitting the answer to each Verifier at the speed of light. The Verifiers each measure the round-trip time and from that obtain an upper bound on how far the Prover can be from them. After the Verifiers compare their findings, they can calculate the position of the Prover using trilateration.

The Verifiers must choose the question in a way that the answer can be calculated under negligible time, and it can only be calculated in possession of all the inputs from all the Verifiers (meaning that if someone has only partial information about the question shares they will fail to guess the correct output – and therefore fail the verification process – with reasonably high probability.)

However, if only classical communication is used, then there is an efficient attack against this scheme [1]. This attack employs multiple coordinating Attackers each of whom intercepts one question share and transmits copies of it to their co-conspirators. This can be done even if the question shares are encrypted since there is no reason why classical data could not be copied.

On the other hand, if the question shares are encoded into qubits then the No-Cloning Theorem provides protection against this type of attack.

2.2 Security against quantum attacks

Quantum position verification is believed to be secure against classical attacks (or to be more precise against all attacks with no pre-shared entanglement [3]). However, it is still vulnerable to a class of quantum attacks – sometimes called teleportation attacks [3].

Moreover, it is been shown that there is a general attack that can be used against all quantum position verification schemes [3, 12]. This general attack employs back-and-forth teleportation between the coordinating Attackers and uses two different types of teleportation. One is the well-known teleportation discovered in 1993 [14], the other is the so-called port-based teleportation [15–17].

This newer teleportation method has multiple potential output ports. When a quantum bit is teleported, the teleportation measurement at the departure side selects one of the potential output ports on the arrival side. The teleported quantum bit appears on the selected port (without the need for any further correction), and random bit-values will appear at all other candidate arrival ports.

This port-based teleportation is also asymptotic [18], meaning there is a chance that the teleportation fails. There is a lower bound on this failure rate and an optimised version of a multi-port scheme has been discovered and published [19].

The fidelity of the quantum state that appears on the correct arrival port is a function of the number of ports, which is, in turn, the function of the number of pre-shared entangled pairs.

This is important because previous works have shown that the amount of entangled qubit pairs needed to ensure a given degree of fidelity (and thus a given probability that the Attackers succeed) scales roughly exponentially as a function of the qubits used to verify the position.

In other words, the attack will fail (with a high degree of probability) if the Attackers are linearly bounded in the amount of pre-shared entanglement.

However, the question remains: is the general attack the best possible attack? We know of specific cases where a position-verification protocol can be more efficiently attacked [3, 20, 21] (e.g. all classical position verification schemes fall in this category.) Moreover, in those cases where we know the optimal attack for a protocol, that attack is either linear or sub-linear [21]. It is currently unclear whether that is the rule or if those optimal attacks were simple enough to be found.

We will assume that there exists at least one position verification protocol which is practically secure – either because the general attack is optimal against them or because the optimal attack poses no significant improvement over the general attack. If this assumption does not hold, the methods we present here do not work either. We will treat this secure position verification protocol as a black box and build our method incorporating that black box.

2.3 Vulnerability to supra-quantum attacks

Recently an interesting connection has been discovered between Popescu–Rohrlich (PR) boxes [22] and position verification [11].

These PR boxes are theoretical devices that can produce extremely strong correlations between distant events. Such strong correlations can be described mathematically, however, our current understanding is that they lie beyond what is achievable in quantum mechanics.

In other words, PR boxes violate the Bell inequalities beyond the Tsirelson bound (for the CHSH inequality this would mean that $|S|$ would go higher than $2\sqrt{2}$) while still conforming to the no-signalling theorem.

Broadbent has shown [11] that if PR boxes were to exist then it would be possible to perform instantaneous non-local quantum computation efficiently, which in turn could be used to break position verification.

However, PR boxes are most likely purely theoretical constructs. An indication of this is that PR boxes have implausible consequences [23]: for example, all distributed Boolean computation could be performed using PR boxes and exchanging only a single bit.

However, we have to admit that we do not have direct evidence that PR boxes do not exist. When we talk about a black box containing a practically secure position verification protocol we merely assume that PR boxes are not real. If that assumption does not hold, then this black box cannot exist either.

3 Access rights

In the following sections, we assume two security levels: a higher, Secure level, and a lower, Public level.

All communications between Verifiers, and all data stored by the Verifiers is at a Secure level. This means that it must be kept confidential and its integrity must be protected. We assume the Verifiers can betrusted to carry this out.

Similarly, the intended Recipient must be at a Secure level and any communication between the Verifiers and Recipient must be through a secure channel.

However, the Prover is not trusted, and in fact, assumed to be an Attacker. Any communication with the Prover and any data handled by the Prover should be considered Public.

Since the Prover is at a Public security level, the Verifiers can no enforce security en route – at least not directly. Accountability of the Prover will be ensured through position verification. This can be done en route before the Prover reaches the Recipient.

4 Position verification as a black box

Let $v$ be the number of Verifiers needed to securely perform a single round of position verification. Security, in this case, means that position verification will fail at a sufficiently high probability in the dishonest case, and – assuming the Prover answers as fast as he can – the uncertainty in the position is acceptably low.

We assume that the position posed by the Verifiers can be securely broken up into $v+1$ number of shares, $v$ of which are transmitted by the Verifiers and an additional one carried by the Prover. (See Fig. 1a.)

Security, in this case, means that if a single Attacker does not possess all the question shares at the claimed position, then any
5 High-level description of quantum tracking

The central idea behind quantum tracking is to hide trackers in some valuable data. The Verifiers require the Prover to prove the position of this data by revealing how to restore one of the trackers from the data, and how to perform position verification using the tracker as an input. The protocol has four steps:

Step 1: Preparing data.
Step 2: Handing over the data to the Prover.
Step 3: Position verification loop.
Step 4: Restoring valuable data.

These steps are detailed in the following subsections.

5.1 Step 1 – preparing data

The Verifiers must prepare the valuable data on a Secure level, ensuring the confidentiality and integrity of that data.

The Verifiers may add some additional data that will help produce the trackers. In Fig. 2, $n$ number of blue memory slots represent this additional data, and the valuable data is represented by $m-n$ number of green memory slots.

If the data is classical, the confidentiality of the data can be ensured by One Time Pad (OTP) encryption, before the data is stored in quantum bits. Integrity can be ensured by adding error detecting code before encryption.

This way, even if the data is moved to a Public security level, its confidentiality is protected.

It is clear that if multiple trackers can be produced from the data, then multiple rounds of position verification can be performed; each verifying that one specific tracker is at point $P$ as the Prover claims.

However, what we want to track is valuable data, and not the trackers. If we were to merely load the trackers and valuable data into the same quantum memory then an Attacker could separate the two and steal the valuable data.

Therefore, we need an operation that ‘combines’ the valuable data and tracker. The valuable and additional data should be combined in a way that the Attackers cannot reliably separate the two without proper instructions from the Verifiers, and any attempt to do so would fail with great probability. In other words, the Verifiers need to prepare the dataset by using a $U_0$ unitary operation that takes the trackers and valuable data as an input and outputs a $\rho_0$ string of quantum bits. (See Fig. 2.)

The Verifiers must continue to enforce confidentiality and integrity during this step, meaning the $U_0$ unitary must remain Secure.

Note that the original, valuable data is not necessarily used during the production of $\rho_0$. Although we introduced the idea of quantum tracking in our previous work [24], this is a recent realisation, which led to developments like the tracking keys that will be introduced in Section 7.

5.2 Step 2 – handing over the data to the Prover

The Verifiers hand over this $\rho_0$ string to the Prover. The data at this point moves to the Public security level. Accountability at his security level will be discussed in the next section.

5.3 Step 3 – position verification loop

During each round of position verification, the Verifiers transmit instructions that describe a $U_i$ unitary transformation (along with the position verification question) that takes the previous $\rho_{i-1}$ state beforehand. The Prover sends $v$ number of answer shares each to a specific Verifier and the additional answer share is merely transmitted to a randomly chosen real Verifier.

From this point forward, we handle this entire process as a black box with a single input: the pre-prepared tracker. (See Fig. 1b) The other question and answer shares are of no importance to us beyond the notion that they exist.

Fig. 1 Secure position verification protocol with $v$ number of Verifiers may be modified for tracking
(a) The question share from an additional (simulated) Verifier can be pre-prepared and used as a tracker, (b) The position verification protocol is then handled as a black box with a single input which is the tracker

Fig. 2 Memory slots filled with trackers and valuable data are combined by an operation $U_0$ producing $\rho_0$ which can be used to restore the $\psi$ trackers one-by-one using $U_i$ or restore the remaining $\varsigma$ trackers and $\phi$ valuable data using $V_i$.

attempt at passing the position verification will fail with a sufficiently high probability.

We will call the additional $(v+1)$th question share a tracker. Later, we will embed this tracker into the data whose position we want to verify.

The tracker must be pre-prepared by the Verifiers at a Secure level, in a way that the Attackers find it unpredictable. This means that the quantum bits are chosen from a large enough set with an appropriate probability distribution (usually this means close enough to uniform probability) and the quantum bits are sufficiently independent from any information accessible by the Attackers during any round of position verification (including independence from other qubits in the question and independence from different rounds of questioning.)

To ensure security, shares must also be large enough to be resilient to the best possible teleportation attack. (This includes the tracker.) All question shares are kept at a Secure level until the position verification question – except the tracker which is handed to the Prover.

Once the Prover assembled the question and calculated the answer, he must break this answer into multiple shares and send them to the Verifiers.

Let us assume the position verification protocol can be modified to include one more (imaginary) Verifier. The actual Verifiers send $v$ number of question shares, and the additional question share that should come from this imaginary Verifier is handed to the Prover.
Producing trackers, even if Verifiers must be able to reveal a final state that ensures that a measurement attempt or copying attempts to decrease integrity, which in turn increases the probability that Prover cannot pass the position verification.

Therefore, the goal of an Attacker would be to pass position verification while either the data is not where she claims, or the integrity of the data is compromised for any reason.

5.4 Step 4 – restoring valuable data

Once the intended Recipient verifies that the data reached him, the Verifiers must be able to reveal a final \( V \) unitary operation, which restores the valuable data and separates it from the remaining trackers so that the valuable data can be decrypted.

\( V \) must be kept at a Secure level, and it should be transmitted to the Recipient on a Secure channel. Once the valuable data is restored, the key of the OTP encryption can be revealed and the integrity can be checked. The assumption that there is no other copy of the valuable data than at the recipient can be reinforced by the fact that Prover passed all position verification steps and the error-correcting code verifies the integrity of the valuable data.

5.5 Additional remarks

It is worth noting that although each tracker should be separable from the rest of the valuable data the tracker might still be entangled with the position verification data or other quantum bits stored by the Verifiers.

The question is what kind of operations can serve this purpose and play the role of \( U_i \) and \( V \).

In the following sections, we propose multiple possible protocols that implement this high-level description.

5.6 Goal of the Attackers

The Attackers’ goal is to hide the fact that the \( \rho \) data is either not where it is supposed to be, or hide that its integrity has been compromised. This violation of integrity could come from copying attempts, measurement attempts, or removal of some part of the data.

6 Shuffling protocol

The shuffling protocol is a simple and straightforward realisation of quantum tracking. It involves partitioning the data into memory slots in a way that the valuable data is indistinguishable from the trackers and performing a random shuffle to hide which memory slot contains trackers and which contains valuable data.

The protocol follows the four steps described in Section 5.

6.1 Step 1 – preparing data

To do this the Verifiers must first prepare the data at a Secure level.

6.1.1 Step 1.1: The classical bit values of the valuable data should be encrypted with OTP encryption (also known as Vernam cipher or OTP for short). This will hide any pattern in the data and make every bit value in the encrypted ciphertext appear to be independent from each other and uniformly chosen to an outside observer. The key to this ciphertext must be held by one or more Verifiers and should not be made public until the final decryption step.

Then the data has to be encoded into quantum bits. If a single tracker consists of \( k \) number of qubits, then the valuable data must also be broken up and encoded into strings of \( k \) qubits. The Verifiers load these strings into memory slots of a quantum storage device and assign a memory address to each slot. For the sake of simplicity, we handle memory addresses as ordinal numbers counting up from 1.

The valuable data must be encoded into qubits in a way that an Attacker cannot tell if a specific memory slot contains tracker or valuable data regardless of the measurement they perform on it. (This means that any attempt to separate them fails with a sufficiently high probability.) This encoding obviously depends on the specific position verification protocol. However, from the point of security, it would be ideal if the unit vectors that the state of the qubit strings were uniformly chosen from all possible states while this choice would be independent of any information an Attacker can access.

6.1.2 Step 1.2: After loading the trackers and valuable data into the quantum memory, the memory slots must be shuffled. (See Fig. 3.). This ensures that the Attackers do not know which is which and by removing any of the memory slots they risk removing a tracker which might be discovered during the position verification.

The shuffling must be done in a way that if there are \( n \) trackers and \( m \) memory slots in total then there is an \( n/m \) chance that any given memory slot contains a tracker. (This random permutation could be produced with a Fisher–Yates shuffle). To ensure the unpredictability of this shuffle only cryptographically secure random number generators should be used.

This random shuffle plays the role of \( U_0 \) in the high-level description. The Verifiers hand the data to the Prover in this encoded and shuffled state. This state is \( \rho_0 \).

6.2 Step 2 – handing over the data to the Prover

The quantum memory which contains \( \rho_0 \) is handed over to the Prover, and is now considered to be at Public security level.
orders serves as operation

hypergeometric probability density as

The Verifiers send these address shares along with the question shares to the Prover. This should be done by using secret splitting, creating \( v \) number of secret shares in a way that all \( v \) shares are necessary to restore the slot address. This is operation \( U_i \) which reveals the memory address of the \( i \)th tracker. The Verifiers send these address shares along with the question shares to the Prover.

6.3 Step 3 – position verification loop

During position verification, the Verifiers reveal the current address of a memory slot which contains a tracker. This should be done by using secret splitting, creating \( v \) number of secret shares in a way that all \( v \) shares are necessary to restore the slot address. This is operation \( U_i \) which reveals the memory address of the \( i \)th tracker. The Verifiers send these address shares along with the question shares to the Prover.

6.4 Step 4 – restoring valuable data

When the data must be decrypted, the Verifiers reveal the addresses and original order of the valuable data. (Restoring the original orders serves as operation \( V' \)).

To do this, the Verifiers must securely store the relationship between the original and current permutation so that they know which memory slot to use during position verification and how to restore the valuable data.

6.5 Probability of detecting irregularities

\( P \) probability stands for the following: after \( r \) rounds of position verification, the Attackers are not detected even if they removed \( q \) number of memory slots. To write \( P \), first, we denote the hypergeometric probability density as \( h(\text{obs}; \text{pop}, \text{att}, \text{draw}) \) where \('\text{pop}'\) is the population size, from which \('\text{att}'\) number of elements have a given attribute and after \('\text{draw}'\) number of draws we observe exactly \('\text{obs}'\) number of elements with the given attribute among those drawn:

\[
h(\text{obs}; \text{pop}, \text{att}, \text{draw}) = \binom{\text{att}}{\text{obs}} \binom{\text{pop} - \text{att}}{\text{draw} - \text{obs}} \binom{\text{pop}}{\text{draw}}
\]

We can write \( P \) as (2)

\[
P = \sum_{j=0}^{q} (A_j \cdot B_j).
\]

where \( A_j \) is the probability that out of the missing \( q \) memory slots exactly \( s \) are trackers, and \( B_j \) is the probability that the Attackers pass the position verification undetected assuming that \( s \) number of trackers are missing. \( A_j \) can be expressed using \( h \) as (3)

\[
A_j = h(\text{obs}, \text{pop}, \text{att}, \text{draw} | j; m, n, q)
\]

whereas \( B_j \) is given by (4)

\[
B_j = \sum_{i=0}^{s} (C_i \cdot p_i).
\]

where \( p_i \) is the probability that the Attackers can pass a single round of position verification assuming the verified tracker is missing. \( C_i \) is the probability, that out of the \( n \) number of trackers – \( s \) of which are missing – the Verifiers select exactly \( t \) missing trackers to be verified during \( r \) rounds of position verification:

\[
C_i = h(\text{obs}, \text{pop}, \text{att}, \text{draw} | i; t, n, s, r)
\]

If \( p_1 \) is small enough, then the probability \( P \) is approximately equal to the chance that none of the missing trackers are used for verification as given by (6)

\[
P \approx \sum_{s=0}^{q} (h(\text{obs}; m, n, q) \cdot h(0; n, s, r)).
\]

Although the shuffling protocol is easy to understand and uses simple operations, it has some drawbacks. First, it would be ideal to have a quantum memory that is filled almost entirely with trackers (\( n/m \approx 1 \)) and if a significant portion (but not all) of the trackers would be used during position verification (\( r/n \approx 1 \) and \( n - r \approx 0 \)).

Another drawback is that every round of position verification confirms the presence of only one single tracker. This means that if there is a valuable memory slot missing, the shuffling protocol will not be able to detect this irregularity.

7 CNOT protocol

Another possible realisation of quantum tracking can be based on a sequence of CNOT operations. The presence of two memory slots can be tested during each round of position verification if the tracker is produced as one of the outputs of a qubitwise CNOT operation.

This method has the following advantage: one of the inputs could be a memory slot containing valuable data therefore the Verifiers have a non-zero chance of detecting partially or completely missing valuable data.

The protocol follows the general steps described in Section 5.

7.1 Step 1 – preparing data

The first step of the CNOT protocol is to prepare data similarly to the shuffling protocol.

7.1.1 Step 1.1: Combining valuable data and tracker starts by shuffling the memory slots in an unbiased way (similarly to the shuffling protocol) meaning that every permutation is equally likely and assuring that neighbours of each tracker slot could be a tracker or valuable data (See Fig. 4).

We will later refer to this new permutation as the logical order, and denote the original contents of the logical \( i \)th memory slot as \( \delta_i \).

7.1.2 Step 1.2: The second step of combining data with the tracker is to perform a sequence of successive qubitwise CNOT operations in a logical order with qubits in the \( i \)th memory slot being the control bits and bits of the \((i + 1)\)th memory slot being the target bits. Then the qubits of the newly modified \((i + 1)\)th memory slot serve as the control bits for the \((i + 2)\)th target slot.

Note that in case of a pure state, it is enough to examine the effect of an XOR operation, since the CNOT maps the orthogonal system of all possible \( \delta_i, \delta_i^+, \in \{0, 1\}^4 \) values onto another orthogonal system:

\[
\text{CNOT}_{i \rightarrow i+1} \sum_{j, \delta_i, \delta_i^+} a(j, \delta_i, \delta_i^+) |j\rangle |\delta_i\rangle |\delta_i^+\rangle = \sum_{j, \delta_i, \delta_i^+} a(j, \delta_i, \delta_i^+) |j\rangle |\delta_i\rangle |\delta_i^+\rangle \oplus |\delta_i^+\rangle.
\]
where $\oplus$ is the XOR, $\gamma \in \{0,1\}^{m-2N}$ represents the rest of the qubits, and $a$ is some probability amplitude depending on $\delta_i$, $\delta_{i+1}$ and $\gamma$.

This means the new $\delta'_i$ contents of $i$th memory slot will be given by (8):

$$\delta'_i = \delta_i \oplus \delta_{i-1} = \delta_i \oplus \delta_{i-1} \oplus \cdots \oplus \delta_i \oplus \delta_1 \oplus \delta.$$  

(8)

Note, that the original $\delta_i$ can be restored by performing another CNOT operation with logically successive memory slots as inputs:

$$\delta'_i \oplus \delta_{i-1} = \delta_i \oplus \delta_{i-1} \oplus \cdots \oplus \delta_i \oplus \delta_1 \oplus \delta.$$  

(9)

7.1.3 Step 1.3: As a next step, the logical order must be hidden from an Attacker by performing another shuffle. We will refer to this third permutation as the physical order. The data has to be stored on a quantum RAM in this physical order before it is handed to the Prover.

These operations together (the first shuffle, consecutive CNOT operations and the second shuffle) constitute $U_0$.

7.2 Step 2 – handing over the data to the Prover

The data is then handed over to the Prover, and is now at a Public security level.

7.3 Step 3 – position verification loop

7.3.1 Step 3.1a: Whenever the Verifiers wish to check the presence of the data they randomly select a tracker (which was not selected previously), they transmit the corresponding physical memory address as well as the physical address of the logically preceding memory slot (using secret splitting). The Prover must perform the CNOT operation which restores the original $\delta_i$ content of the tracker and leaves the logically preceding $\delta'_{i-1}$ unchanged. This is the $U_i$ operation.

7.3.2 Step 3.1b: However, there is an interesting case we must examine more closely. In case the preceding $\delta'_{i-1}$ memory slot is missing, this operation cannot be performed. This happen if the missing memory slot is also a tracker and it was used during a previous round of position verification. Moreover, there could be $j$ number of logically preceding tracker slots (from $i-j$ to $i-1$) each of which is missing. (Details are shown in Fig. 5.)

In this case, the $U_i$ operation takes a slightly different form. The Verifiers handle this situation by preparing an extra $\epsilon_j$ qubit string and transmitting it alongside the position verification information. This extra information can be prepared assuming the Verifiers know the original $(\delta_{i-j}$ to $\delta_{i-1})$ value of these trackers. Since these values were originally chosen and encoded by them, it is merely a question of proper administration. With this $\epsilon_j$ can be calculated as (10)

$$\epsilon_i = \delta_{i-1} \oplus \delta_{i-2} \oplus \cdots \oplus \delta_{i-j}.$$  

(10)

The Prover can restore the original $\delta_i$ value from $\delta'_i$, $\epsilon_i$ and the first non-missing $\delta_{i-1}$ as:

$$\delta_i = \delta'_i \oplus \epsilon_i \oplus \delta_{i-1} = \delta_i \oplus \delta_{i-1} \oplus \delta.$$  

(11)

7.4 Step 4 – restoring valuable data

The valuable information can be completely restored if the Verifiers reveal the original order of memory slots, and transmit all necessary $\epsilon_j$ states. With this information operation $V$ can be performed by repeating the consecutive CNOT operations, substituting $\epsilon_j$ states for the missing trackers.

7.5 Leaked information during tracking

Whenever the Verifiers reveal an $i$ memory address, the Attackers gain some information. The following question may arise: Does this leaked information weaken security, i.e. could the Attackers use this information to optimise their attack? We claim that they cannot.

To prove this, first, let us examine how the Verifiers could detect that a given $i$th memory slot is missing. There are two possibilities. (i) $i$th memory slot is a tracker and the Verifiers select it for position verification – we will refer to this as the $i$th slot being selected. (ii) The (logically) consecutive $i+1$st memory slot is a tracker and it is selected for position verification. Therefore, the $i$th memory slot in question must be revealed as a control slot for the CNOT operation – we will refer to slots like this as revealed.

These two mechanisms can exist even if the $i$th slot was revealed before since it still has to be revealed again to select the first logically consecutive non-missing slot. (This is due to the $\epsilon_i$ states and additional information described in Section 7.3.2 Step 3.1/b.) So there are still two mechanisms for detecting the irregularity as long as that memory slot did not become the $m$th: the current logical last.
However, for the last, m′th memory slot (in logical order) there is no next. For that reason it would be optimal for the Attackers to remove that last memory slot, which is protected by only a single mechanism and therefore the irregularity has a lower probability of being detected.

From the Attackers point of view, there is no difference between the memory slots that have not been revealed yet. Any single one of them is equally likely to be the last slot, and any single one of them is equally likely to be a tracker.

Similarly, there is no difference between the revealed trackers, because the shuffle ensures that all permutations are equally likely. Moreover, this holds true even after trackers are selected and removed: if any element of an unbiased permutation is removed, the remaining elements still form an unbiased permutation. Therefore, each memory slot is equally likely to be anywhere in a logical order – even if \( j \) number of consecutive trackers have been selected before them.

Firstly, this means that the probability that a revealed memory slot is a tracker is equally likely for all revealed slots: \( n′/n′ \) where \( n′ \) is the number of not selected trackers and \( m′ \) is the number of not selected memory slots. Secondly, the probability that the revealed memory slot became the last is also equally likely for each memory slot (again due to the fact that every permutation is equally likely and trackers are selected in a random order.)

Therefore, the only question for the Attackers is whether to remove one of the revealed memory slots or choose randomly from among those that were not revealed. Let us assume that there are \( a \) number of revealed memory slots and \( b \) number of not revealed memory slots out of the currently \( m′ \) number of not selected slots.

Since every permutation is equally likely and trackers are selected randomly, the \( p(a) \) probability that a revealed slot is currently the logical last is \( a/m′ \). However, the set of revealed trackers has a number of indistinguishable elements in it, each with equal probability to be the last. Assuming the last slot has been revealed, this \( p(\text{not} a) \) conditional probability is \( 1/a \).

With this the probability that the \( i \)th revealed memory slot is the last is given by (12) for all \( i \):

\[
p(i \cap a) = p(a) \cdot p(\text{not} a) = \frac{a}{m′} \cdot \frac{1}{a} = \frac{1}{m′}.
\]

Similarly, the probability for the \( j \)th not yet revealed memory slot to be the last is given by (13) for all \( j \):

\[
p(i \cap b) = p(b) \cdot p(\text{not} b) = \frac{b}{m′} \cdot \frac{1}{b} = \frac{1}{m′},
\]

where \( p(b) \) is the probability that the current logically last slot has not been revealed and \( p(\text{not} b) \) is the conditional probability that the \( j \)th slot is the last from among those not revealed assuming the currently last slot has not been revealed.

Therefore, revealing memory slots gives no actionable information to the Attackers. This, however, does not mean that there are no other attack strategies they could follow: for example, they could wait until the last moment when the number of trackers is the lowest to remove a memory slot.

8 Tracker keys

Previously we proposed protocols where the valuable data is encrypted and stored along with randomly chosen trackers. In this section, we propose a slightly different approach: to use the remaining trackers to produce an encryption key once they arrive at their destination.

Here we use a combination of quantum tracking and quantum key distribution [25]. This requires a position verification protocol whose input qubit string can also be used to generate a random key.

As an example, let us use the BB84 protocol to generate a key. Furthermore, let us assume that the valuable data is classical and trackers are separable states that do not need to be entangled with question shares of the position verification protocols or data stored by the Verifiers. Whether these assumptions hold, depends largely on the secure position verification protocol.

8.1 Step 1 – preparing data

8.1.1 Step 1.1: First, the Verifiers choose \( k \) number of classical random bit values, and encode them into qubits randomly, choosing between rectilinear and diagonal bases.

They produce \( n \) number of qubit \( γ_i \) strings this way, ensuring that \( n \) is large enough that even after the necessary rounds of position verification and production of a refined key this key has enough bits to encrypt the secret message using OTP.

8.1.2 Step 1.2: Then for each \( γ_i \) qubit string the Verifiers choose a random \( T_i \) rotation in the Hilbert space to produce a \( δ_i \) state. All these \( T_i \) rotations collectively form \( U_0 \) and therefore they must be stored by the Verifiers at a Secure level, and must be chosen in a way so that they are hard to guess for Attackers.

8.2 Step 2 – handing over the data to the Prover

Then the Verifiers store these \( δ_i \) states on a quantum memory, which they hand over to the Prover. This state is now considered to be at Public security level.

8.3 Step 3 – position verification loop

During transport, the Verifiers can track the dataset by using up some of the memory slots for position verification. This is done by calculating and revealing a \( U_i \) rotation, which transforms \( δ_i \) into a \( δ_i \) tracker state. (These \( δ_i \) tracker states are pre-agreed by the Verifiers at a Secure level and must be hard to guess for Attackers without knowing \( U_i \))

If \( δ_i \) is a separable state (which we assumed it is), then \( U_i \) is a rotation. This rotation is revealed using secret splitting and shares are transmitted alongside the position verification data.

Missing trackers can even be replenished by transmitting new trackers (prepared by the Verifiers as described in Step 1) to replace those selected for verification.

8.4 Step 4 – restoring valuable data

Once the data reaches the Recipient, the Verifiers reveal how to restore the original \( γ_i \) qubit strings, by revealing the \( T_i^{-1} \) inverse rotations. These rotations collectively form the restoring \( V \) operation.

The resulting strings can be used as an input for the BB84 protocol, producing a shared secret key, which in turn can be used by the Verifiers to encrypt the valuable data using OTP.

The Verifiers can transmit this ciphertext through a public channel to the Recipient. The shared key is then used to restore the valuable data.

8.5 Additional remarks

An interesting feature of tracking keys is that the valuable data itself was not used to produce the \( δ_i \) states. However, once the shared secret key is generated using that key to generate a cipher text is essentially indistinguishable from secret splitting (with the key and cipher text being two shares of the same plaintext.) Since secret splitting is symmetrical, it can be argued that from the point of view of an Attacker the transmitted key is the encrypted message and the cipher text produced by the Verifiers can be considered the key.

A possible use case for tracker keys would be to locate the Attackers during a denial of service attack. To do this, nodes in a quantum network must have position verification capability (the network itself playing the role of the Prover.) This would hinder the attack since if the Attackers would gain control of a single node (or edge), position verification would fail at the next node in the path revealing their location.

If Attackers would gain control of multiple nodes in a sub-path they could use their own layer of encryption which would render the key useless to the sender Alice and recipient Bob while they would be able to pass position verification. However, they would still not be able to lie about the path itself nor about the first node
which is not under their control. (This last node could be Bob himself.)

This could considerably weaken the attack since communication could be restored by following the sub-path backwards starting from the first node that the Attackers did not control and gradually regaining control over each edge and node. Depending on the topology this can be significantly easier for Alice and Bob than searching all nodes and edges along all possible paths.

(If the key follows a walk instead of a path, this can be generalised.)

8.6 Goal of the Attackers

Here, the Attackers have an additional goal of gaining information about the shared key. This is prevented through the quantum key distribution protocol performed in Step 4.

Even if the Attackers were to gain access to all $T^\nu_i$ inverse rotations (which would violate the security policies), they would still have to break the key distribution protocol.

9 Conclusions

Position verification can be used to verify that a distant communicating party is where they say they are. Assuming that a practically secure position verification protocol exists, it can be used to track valuable data. In this paper, we gave a general description of how this can be achieved and proposed three protocols to implement this general scheme: the shuffling protocol, the CNOT protocol and tracker keys.

We examined whether the information revealed during position verification in the CNOT protocol allows Attackers to optimise their attack strategies and conclude that it does not.

9.1 Comparing protocols

The shuffling protocol is the simplest and requires little to no quantum operations from the Prover beyond the position verification.

However, it has some drawbacks. The presence of only one memory slot is checked each time, and if the integrity of valuable data the protocol has no chance of detecting that irregularity.

The CNOT protocol is more computationally intensive but grants the possibility of detecting if the integrity of valuable data has been violated.

Tracker keys would be the most flexible. Since any memory slot could be transformed into a tracker though $U_T$ or into shared key through $T^\nu_i$, there would always be a chance of detecting irregularities of any memory slot.

However, the integrity of memory slots is verified one by one. It would be better if more than one intact memory slot should be present to produce a tracker. Moreover, the protocol we described assumes the tracker can be a separable state. This depends on the secure position verification and may not hold.

9.2 Open questions

A central open question is whether a practically secure protocol for position verification exists. Although still very possible it is yet unproven.

Furthermore, the black-box model of position verification we presented in Section 4 assumes that the number of Verifiers can be increased by one. This is a likely assumption, but whether it actually holds or not, is a hard answer without knowing how the position verification works.

It is also unclear whether tracker keys are possible if the tracker needs to be entangled with additional data. If that is the case, we have no proof that there is a secure way to produce the tracker without compromising the position verification process by making easier for the Attackers to guess the answer for the position verification question without possessing all question shares.

Questions regarding the security of the proposed protocols should be studied further.

10 Acknowledgments

This research was supported by the National Research Development and Innovation Office of Hungary within the Quantum Technology National Excellence Program (Project No. 2017-1.2.1-NKP-2017-00001). The research is connected to COST Action CA15220 Quantum Technologies in Space.

11 References

[1] Chandran, N., Goyal, V., Moriarty, R., et al.: 'Position based cryptography'. Advances in Cryptology – CRYPTO 2009, Santa Barbara, USA, August 2009, pp. 391–407

[2] Kent, A., Munro, W.J., Spiller, T.P.: ‘Quantum tagging: authenticating location via quantum information and relativistic signaling constraints’, Phys. Rev. A, 2013, 84, (1), pp. 150–179

[3] Buhrman, H., Chandran, N., Fehr, S., et al.: ‘Position-based quantum cryptography: impossibility and constructions’, SIAM J. Comput., 2013, 43, (1), pp. 337–351

[4] Broadbent, A., Schaffner, C.: ‘Quantum cryptography beyond quantum key distribution’, Des. Codes Cryptogr., 2016, 382, (78), pp. 351–382

[5] Qi, B., Siopis, G.: ‘Loss-tolerant position-based quantum cryptography’, Phys. Rev. A, 2015, 91, (4), p. 042337

[6] Kent, A.: ‘Quantum tagging for tags containing secret classical data’, Phys. Rev. A, 2011, 84, (2), p. 022335

[7] Malaney, R.: ‘The quantum cat’, IEEE Wirel. Commun. Lett., 2016, 5, (6), pp. 624–627

[8] Unruh, D.: ‘Quantum position verification in the random oracle model’. Advances in Cryptology – CRYPTO 2014, Santa Barbara, USA, August 2014, pp. 1–18

[9] Imre, S., Gyöngyösi, L.: ‘Advanced quantum communications – an engineering approach’ (Wiley & Sons, New Jersey, USA, 2012)

[10] Speelman, F.: ‘Instantaneous non-local computation of low T-depth quantum circuits’. 11th Conf. on the Theory of Quantum Computation, Communication and Cryptography (TQC 2016), Berlin, Germany, September 2016, pp. 9:1–9:24

[11] Broadbent, A.: ‘Popescu-Rohrlich correlations imply efficient instantaneous nonlocal quantum computation’, Phys. Rev. A, 2016, 94, (2), p. 022318

[12] Beigi, S., König, R.: ‘Simplified instantaneous non-local quantum computation with applications to position-based cryptography’, New J. Phys., 2011, 13, (9), p. 093036

[13] ‘Position-based quantum cryptography for multi-located prover and single verifier’. Available at http://2017.acyrypt.net/wp-content/uploads/2017/09/Waf787 pdf, accessed 15 January 2020

[14] Bennett, C.H., Brassard, G., Crépeau, C., et al.: ‘Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels’, Phys. Rev. Lett., 1993, 70, (13), p. 1895

[15] Ishizaka, S., Hiroshima, T.: ‘Asymptotic teleportation scheme as a universal programmable quantum processor’, Phys. Rev. Lett., 2008, 101, (24), p. 240501

[16] Ishizaka, S., Hiroshima, T.: ‘Quantum teleportation scheme by selecting one of multiple output ports’, Phys. Rev. A, 2009, 79, (4), p. 042306

[17] Studziński, M., Strelchuk, S., Mozrzymas, M., et al.: ‘Port-based teleportation in arbitrary dimension’, Sci. Rep., 2017, 10871, (7), pp. 1–11

[18] Christandl, M., Ledizky, F., Majenz, C., et al.: ‘Asymptotic performance of port-based teleportation’, BIRS workshop, Banff, Canada, July 2019. Available at https://arxiv.org/abs/1809.10751, accessed April 2020

[19] Mozrzymas, M., Studziński, M., Strelchuk, S., et al.: ‘Optimal port-based teleportation’, New J. Phys., 2018, 20, (5), p. 053006

[20] Chakraborty, K., Leverrier, A.: ‘Practical position-based quantum cryptography’, Phys. Rev. A, 2015, 92, (5), p. 052304

[21] Ribeiro, J., Grosshans, F.: ‘A tight lower bound for the BB84-states quantum-position-verification protocol’. Available at https://arxiv.org/abs/1504.07171, accessed 20 April 2020

[22] Popescu, S., Rohrlich, D.: ‘Quantum nonlocality as an axiom’, Found. Phys., 1994, 24, (3), pp. 379–385

[23] van Dam, W.: ‘Implausible consequences of super-strong nonlocality’, Nat. Comp., 2013, 12, (1), pp. 9–12

[24] Galambos, M., Bacsardi, L.: ‘Using a quantum tracker to verify the geographical position of a dataset’. IAC-2019, Washington D.C., USA, October 2019, pp. 1–9

[25] Gyongyosi, L.: ‘Multicarrier continuous-variable quantum key distribution’, Theor. Comput., 2020, 816, pp. 67–95

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)