ARTICLE OPEN

Foiling covert channels and malicious classical post-processing units in quantum key distribution

Marcos Curty[1] and Hoi-Kwong Lo[2]

The existing paradigm for the security of quantum key distribution (QKD) suffers from two fundamental weaknesses. First, covert channels have emerged as an important threat and have attracted a lot of attention in security research in conventional information and communication systems. Covert channels (e.g. memory attacks) can fatally break the security of even device-independent quantum key distribution (DI-QKD), whenever QKD devices are re-used. Second, it is often implicitly assumed that the classical post-processing units of a QKD system are trusted. This is a rather strong assumption and is very hard to justify in practice. Here, we propose a new paradigm for the security of QKD that addresses these two fundamental problems. Specifically, we show that by using verifiable secret sharing and multiple optical devices and classical post-processing units, one could re-establish the security of QKD. Our techniques are rather general and they apply to both DI-QKD and non-DI-QKD.

npj Quantum Information (2019)5:14 ; https://doi.org/10.1038/s41534-019-0131-5

INTRODUCTION

There has been much interest in the subject of quantum key distribution (QKD) in recent years because it holds the promise of providing information-theoretically secure communications based on the laws of quantum physics.[1,2] There is, however, a big gap between the theory[3,4] and the practice[5–8] of QKD, and the security of QKD implementations is seriously threatened by quantum hacking.[9–13] To solve this problem, the ultimate solution is device-independent (DI)-QKD,[14–17] which allows the legitimate users of the system (typically called Alice and Bob) to treat their quantum devices as "black boxes". The security of DI-QKD is based on a loophole-free Bell test.[18,19] Although no experimental implementation of DI-QKD has been realised yet, the recent demonstrations of loophole-free Bell tests[20–24] might bring DI-QKD closer to experimental realisation.

Despite its conceptual beauty, DI-QKD is however not foolproof. Indeed, one cannot expect that all QKD users will have expertise in experimental quantum optics and electronics. So, unless Alice and Bob manufacture their own QKD devices themselves, it could be very hard for them to guarantee that the QKD "black boxes" bought from a vendor are indeed honest, as it is assumed in the security proofs. For instance, it was shown in ref. [25] that DI-QKD is highly vulnerable to the so-called memory attacks, where a hidden memory device (planted by the eavesdropper, Eve, in say Alice's setup) stores up the key material generated in each QKD session and then leaks this information to Eve in subsequent QKD runs. This situation is illustrated in Fig. 1. Obviously, this is a fatal loss of security for DI-QKD. Whenever a QKD system is reused for subsequent QKD sessions, the security of the keys generated in previous QKD runs might be compromised. Note that this is particularly problematic in a network setting with multiple users (who may not all be trustworthy) due to the impostor attack.[25]

Moreover, we remark that, in principle, memory attacks also work against non-DI-QKD. This is so because, in practice, it could be quite challenging to check whether or not a purchased QKD setup contains such memory. In the following, whenever we refer to a QKD system, it will be implicitly understood that it could be either DI-QKD or non-DI-QKD; our results apply to both frameworks.

The existence of memory attacks in DI-QKD shows that quantum mechanics alone is not enough to guarantee security. Indeed, in the presence of malicious devices, the resulting secret key is fundamentally insecure due to causality. The only reliable way to assure that covert communication is *not* happening is for the legitimate QKD users and Eve to be space-like separated. Once a key is generated in a QKD session, it is a classical object and thus is subject to copying. Once copied, QKD modules and classical post-processing units have plenty of time and opportunities to covertly leak the key to Eve.

More generally, covert channels[26] have attracted massive attention in conventional security of computing and communication systems.[27–29] With covert channels, seemingly innocent communications in a protocol could leak crucial information that is fatal to its security. Indeed, there are plenty of covert ways of leaking information to Eve in QKD. Even if Alice and Bob's devices are shielded in a Faraday's cage, it could be hard for them to find a perfect Faraday's cage in practice. In fact, industrial-grade electromagnetic shielding can provide only about say 100 dB shielding for a certain wavelength range of electromagnetic emissions.[30] Also, these shields are ineffective against covert channels that exploit for example low-frequency magnetic fields, low-energy acoustic or ultrasonic emissions, gamma rays, neutrons, not to mention multiplexing a covert signal onto the device's unavoidable electricity consumption and heat generation. Note that memory attacks are simply one example of covert channels. One main motivation of our work is indeed to find a new and general paradigm of security that can reduce the risk due to

[1]Escuela de Ingeniería de Telecomunicación, Department of Signal Theory and Communications, University of Vigo, E-36310 Vigo, Spain and [2]Center for Quantum Information and Quantum Control, Department of Physics and Department of Electrical & Computer Engineering, University of Toronto, Toronto M5S 3G4, Canada
Correspondence: Marcos Curty (mcurty@com.uvigo.es)

Received: 2 July 2018 Accepted: 16 January 2019
Published online: 06 February 2019

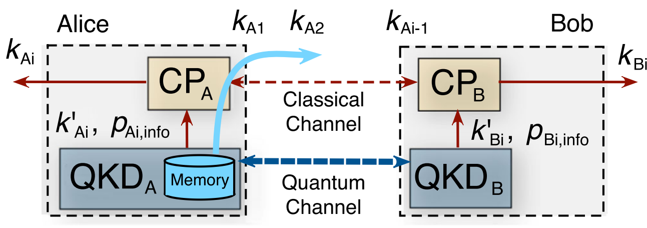

Fig. 1 Schematic representation of a memory attack. In the *i*th QKD run, Alice and Bob's QKD modules, $\mathrm{QKD_A}$ and $\mathrm{QKD_B}$, use a quantum channel to produce a raw key, k'_{Ai} and k'_{Bi}, as well as certain protocol information, $p_{Ai,\mathrm{info}}$ and $p_{Bi,\mathrm{info}}$, respectively. The content of $p_{Ai,\mathrm{info}}$ and $p_{Bi,\mathrm{info}}$ depends on the particular QKD protocol implemented. For instance, in the standard decoy-state BB84 scheme,[81–83] $p_{A,\mathrm{info}}$ contains the basis and the decoy setting information per emitted signal, while $p_{B,\mathrm{info}}$ contains Bob's measurement basis and it also indicates which signals produced a click in his measurement apparatus. The raw key and the protocol information is sent to Alice and Bob's classical post-processing units, $\mathrm{CP_A}$ and $\mathrm{CP_B}$, which are connected via an authenticated classical channel. These units generate a secret key, k_{Ai} and k_{Bi}, by applying various post-processing techniques. In a memory attack, Eve hides a memory in say Alice's module $\mathrm{QKD_A}$ to first store up the key material generated in each QKD run and then leak this information to her by hiding it in say the decision of abort or not abort of a subsequent QKD session. For example, if a particular bit value, say the *j*th bit, of the key generated in the first QKD run is 0, then this memory makes that a permuted *σ*(*j*)th QKD run aborts. (Here, *σ* is a permutation and *σ*(*j*) is the permuted value of *j*.) This could be achieved, for instance, by outputting a raw key with a high quantum bit error rate (QBER). And, if the *j*th bit value of the key is 1, then the *σ*(*j*)th QKD run does not abort. That is, by simply learning whether or not the *σ*(*j*)th QKD round has aborted, Eve could obtain the *j*th bit value of the first QKD session key. Alternatively, Eve might also leak the key material produced in a certain QKD round by simply hiding it in the public discussion of subsequent QKD runs. We refer the reader to ref. [25] for more details. Memory attacks are a fundamental threat to the security of DI-QKD

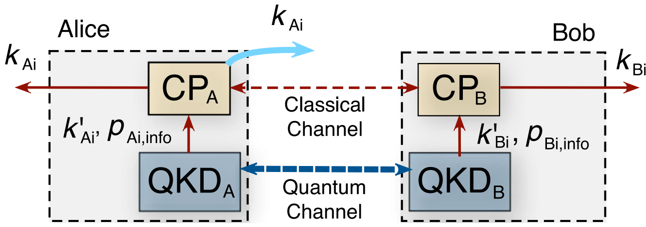

Fig. 2 Schematic representation of a hardware/software Trojan Horse attack. In QKD, it is commonly and implicitly assumed that the classical post-processing units $\mathrm{CP_A}$ and $\mathrm{CP_B}$ are trusted. However, due to the many hardware[31–34] and software[35] Trojan Horse attacks that have been performed against conventional cryptographic systems, this trust is a very strong and unjustified assumption. For instance, Eve could modify a chip, or infect the software with malware, to make it fail at a crucial time, or to hide a backdoor in say Alice's unit $\mathrm{CP_A}$ that leaks the final key, k_{Ai}, generated in say the *i*th QKD run to her.[69,70] Due to the complexity and fabrication costs of current chips, these devices are typically designed by different parties, manufactured by an external foundry, and packaged and distributed by separate companies. This gives Eve multiple opportunities to meddle with the hardware. More importantly, hardware Trojans can be very hard to detect in practice. This is so because even slight adjustments to the electrical properties of a few transistors (from the billions of them contained in today's chips) could already compromise the security. Also, Eve could easily bypass post-fabrication tests by crafting attack triggers that require a sequence of unlikely events, or by altering only a subset of the chips in question.[32,33] Similar arguments apply as well to software malware. This shows the weaknesses of the classical post-processing units. We refer the reader to the caption of Fig. 1 for the meaning of the different elements in this figure

not only memory attacks, but also due to *any* type of covert channels in QKD.

Another key weakness in standard QKD security proofs is that they all implicitly assume that the classical post-processing units are trusted. These units are supposed to distil a secure secret key from the raw data generated by the QKD modules by applying techniques, such as post-selection of data (or so-called sifting), parameter estimation, error correction, error verification and privacy amplification. However, in view of the many hardware[31–34] and software[35] Trojan Horse attacks that have been performed recently in conventional cryptographic systems, such trust is a very strong and unjustified assumption. This scenario is illustrated in Fig. 2. Hardware and software Trojans constitute today a key threat to the security of conventional cryptographic devices[34,36] and this threat is expected to only rise with time, so it cannot be neglected when analysing the security of a QKD implementation.

In summary, the current paradigm for the security of QKD essentially relies on Alice and Bob trusting *all* their devices including both their quantum communication components and classical post-processing units and all ancillary components, such as, for example, power regulators. Since even trusted vendors often do not build their chips and power regulators themselves, but rely on secondary and tertiary supplies, it is a fact of life that no one could possibly check and verify the security of all components in QKD. Even the US military does not have the time or capability to check the security of every component used in its hardware.[31,37] How could one expect any vendor to have such a capability? This seems like an impossible situation for the security of practical QKD systems. So, here comes the key question: How do we address covert channels and prove security in QKD with untrusted classical post-processing units?

In this paper, we address the above key question directly. We propose a simple, but important paradigm shift in the field of QKD. Instead of the standard approach of buying a QKD "black-box" from vendors (which is purged by many aforementioned real-life security pitfalls due to covert channels and untrusted classical post-processing units), we propose that it is important for Alice and Bob to employ redundancies.

For this, we draw inspiration from classical error correction via redundancy, where one can correct up to a certain amount of errors by adding enough redundant information to the messages. In so doing, it is possible to achieve reliable data transmission through noisy channels given that the noise is not too high. Analogously, due to the difficulty of verifying whether or not a device can be trusted, we might assume that, for Eve, it might be more difficult to tamper with several devices from different vendors than corrupting just one QKD device from a single vendor. While this is an assumption, it is probably the best Alice and Bob can hope for, as it is clear that if they cannot trust any of their devices, no security is possible, while to simply trust all the devices is very risky. Then, assuming that the number of corrupted devices is limited, and given that Alice and Bob have enough redundant devices, it might be possible for them to achieve secure secret key distribution.

More formally, we use the idea of secure multiparty computation.[38–41] In conventional cryptography, an important question is how to achieve unconditional security in the presence of cheaters. Clearly, if everyone is a cheater, security is impossible to achieve. However, it is still good to achieve security when the number of cheaters is bounded. Verificable secret sharing (VSS)[42,43] is an important primitive in multiparty secure computation. It allows a dealer to split a message between several parties such that only authorised groups of parties can collaborate and reconstruct the message. Importantly, VSS ensures that even if the dealer is malicious there is a well-defined message that the parties can later reconstruct (see Methods section). We remark that secret sharing (SS) is commonly deployed in most modern hardware secure

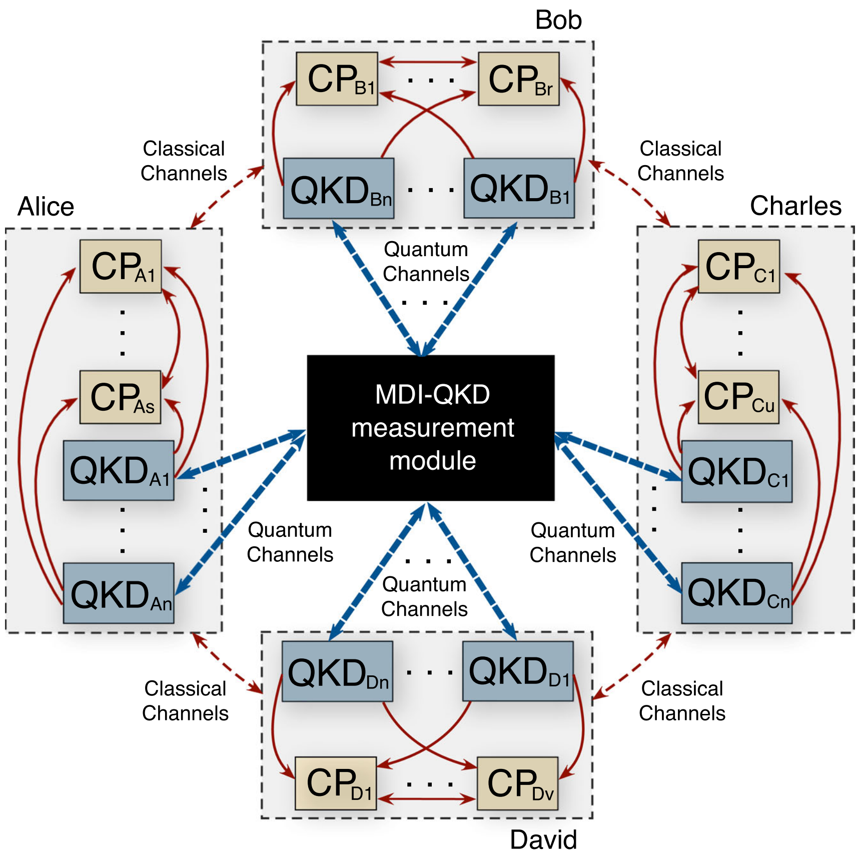

Fig. 3 Schematic representation of a MDI-QKD network with multiple QKD transmitter modules and classical post-processing units. Note that the measurement devices are often the most expensive components of an entire QKD system because single photon detection is highly non-trivial. MDI-QKD[47] allows measurement modules to be totally untrusted, which means that there is no need for redundant measurement modules if our proposal is employed with MDI-QKD. The users just need to have multiple transmitters and classical post-processing units, which thanks to the development of cheap chip-based QKD systems[52–54] and low-cost laptops, we believe, could render our proposal cost effective in the future. We remark that our approach is also fully compatible with quantum relays and quantum repeaters

modules,[44–46] and is the current strategy employed to stop hardware Trojans in conventional hardware design.[31] Since the QKD application is supposed to provide the strongest link of security, our view is that it is very natural for QKD to employ ideas in SS. Also, we note that the use of SS schemes to split the secret information in shares seems to be unavoidable to prevent the malicious devices to have full information about the key to begin with.

The price that we pay is that now Alice and Bob have to use a redundant number of QKD modules and classical post-processing units. Fortunately, however, with the recent development of measurement-device-independent QKD (MDI-QKD),[47–51] and chip-based QKD,[52–54] as well as low-cost laptops, it is reasonable to expect that the cost of QKD modules and classical post-processing units might decrease dramatically over time (see Fig. 3). So, it is not unrealistic to consider that each of Alice and Bob could possess a few QKD modules and classical post-processing units, each of them purchased from a different vendor.

Moreover, as the interest in building a Quantum Internet[55,56] grows exponentially over time, more quantum networks are currently built in the world.[57–62] In a quantum network, it may be natural that two users are connected by multiple QKD paths. In this case, it is rather natural to employ SS schemes to enhance communication security in quantum networks.

Now, provided that the majority of the vendors (or QKD paths) are honest and careful in the manufacturing of their devices, it might not be entirely unreasonable to assume that at least one pair of QKD modules (or QKD paths) is honest/reliable and the number of malicious classical post-processing units is strictly less than one-third of the total number of them. Like in the classical error correction scenario, in standard conventional cryptography it is up to Alice and Bob to determine their security policy and to decide in advance how many corrupt parties they want to be secure against.

With these assumptions in place, we can then prove security in different QKD scenarios with malicious devices by applying privacy amplification techniques[63] in combination with VSS[42,43] from secure multiparty computation[38] in conventional cryptography.[39–41] Importantly, if we disregard the cost of authenticating the classical communications, our protocols are optimal with respect to the resulting secret key rate. Moreover, the operations involved are based on simple functions in linear algebra, such as bit-wise XOR and multiplication of matrices. So, they are conceptually simple and easy to implement.

We remark that naive solutions based on directly taking the XOR between various keys suffer from two main drawbacks, as we show later. First, they cannot guarantee the correctness of the resulting key. And, second, if the classical post-processing unit that performs this operation is malicious, the secrecy of the key is not guaranteed either. In contrast to this naive approach, our solution *can* guarantee both the correctness and the secrecy of the final key.

RESULTS

Let us start by describing in more detail the general scenario that we consider. It is illustrated in Fig. 4a. Alice and Bob have n pairs of QKD modules, and say Alice (Bob) has s (r) classical post-

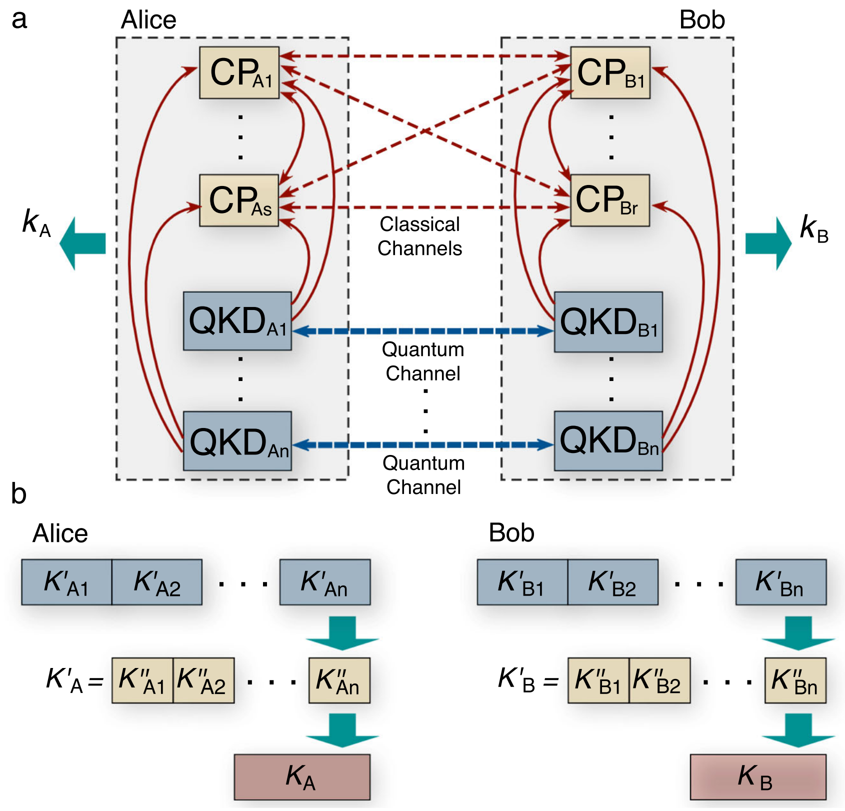

Fig. 4 Schematic representation of the general scenario considered. **a** A QKD setup with multiple QKD modules and classical post-processing units. We assume that up to $t < n$ pairs of QKD modules, up to $t' < s/3$ units $\mathrm{CP}_{\mathrm{A}i'}$ and $t'' < r/3$ units $\mathrm{CP}_{\mathrm{B}i''}$ could be corrupted. The goal is to distil shares of an ε-secure key, k_A and k_B. In the figure, the thin red solid lines represent secure classical channels, the thin red dashed lines denote authenticated classical channels, and the blue thick dashed lines are quantum channels. **b** General strategy to distil shares of k_A and k_B. First, each pair $\mathrm{QKD}_{\mathrm{A}i}$ and $\mathrm{QKD}_{\mathrm{B}i}$ outputs a raw key, $k'_{\mathrm{A}i}$ and $k'_{\mathrm{B}i}$, together with the protocol information, and sends them to the CP units at Alice and Bob's side, respectively. From $k'_{\mathrm{A}i}$ and $k'_{\mathrm{B}i}$, these units distil a supposedly $(\varepsilon_{\mathrm{cor}}/n)$-correct and $(\varepsilon_{\mathrm{sec}}/n)$-secret key, $k''_{\mathrm{A}i}$ and $k''_{\mathrm{B}i}$, and then concatenate these keys to form $k'_\mathrm{A} = [k''_{\mathrm{A}1}, k''_{\mathrm{A}2}, \ldots, k''_{\mathrm{A}n}]$ and $k'_\mathrm{B} = [k''_{\mathrm{B}1}, k''_{\mathrm{B}2}, \ldots, k''_{\mathrm{B}n}]$. Finally, the CP units apply privacy amplification to k'_A and k'_B to remove the information held by the corrupted QKD modules and obtain k_A and k_B. In the presence of corrupted CP units, all these steps are realised in a distributed setting by acting on data shares generated with a VSS scheme

processing units at their disposal. Alice's modules $\mathrm{QKD}_{\mathrm{A}i}$, with $i = 1,\ldots,n$, are connected to the classical post-processing units $\mathrm{CP}_{\mathrm{A}i'}$, with $i' = 1,\ldots,s$, via secure channels (i.e., channels that provide both secrecy and authentication). Also, all the units $\mathrm{CP}_{\mathrm{A}i'}$ are connected to each other via secure channels. The same applies to Bob. Importantly, since all these secure channels are located only within Alice and Bob's labs, in practice they could be implemented, for instance, by using physically protected paths (e.g., physical wires that are mechanically and electrically protected against damage and intrusion) which connect only the prescribed devices. Furthermore, each $\mathrm{QKD}_{\mathrm{A}i}$ is connected to its partner $\mathrm{QKD}_{\mathrm{B}i}$ via a quantum channel, and each $\mathrm{CP}_{\mathrm{A}i'}$ is connected to all $\mathrm{CP}_{\mathrm{B}i''}$, with $i'' = 1,\ldots,r$, via authenticated classical channels.[64,65]

Moreover, for simplicity, we shall consider a so-called threshold active adversary structure. That is, we will assume that up to $t < n$ pairs of QKD modules, up to $t' < s/3$ units $\mathrm{CP}_{\mathrm{A}i'}$ and up to $t'' < r/3$ units $\mathrm{CP}_{\mathrm{B}i''}$ could be corrupted. We say that a pair of QKD modules is corrupted when at least one of them is corrupted. Also, we conservatively assume that corrupted devices do not necessarily follow the prescriptions of the protocol but their behaviour is fully controlled by Eve, who could access all their internal information. We refer the reader to the Supplementary Notes 3 and 4 for the definition of general mixed adversary structure[66] and the security analysis of QKD against this type of general adversary. Also, we note that less conservative adversarial models like e.g., those studied in ref. [25] might be also valid in certain scenarios.

The final goal is to generate a composable ε-secure key, k_A and k_B. That is, k_A and k_B should be identical except for a minuscule probability $\varepsilon_{\mathrm{cor}}$, and say k_A should be completely random and decoupled from Eve except for a minuscule probability $\varepsilon_{\mathrm{sec}}$, with $\varepsilon_{\mathrm{cor}} + \varepsilon_{\mathrm{sec}} \le \varepsilon$.[67,68] Importantly, since now some QKD modules and classical post-processing units could be corrupted, the secrecy condition also implies that k_A and k_B must be independent of any information held by the corrupted devices *after* the execution of the protocol. Otherwise, such corrupted devices could directly leak that information to Eve. Obviously, at the end of the day, some parties might need to have access to the final key, and thus one necessarily must assume that such parties are trusted and located in secure labs. In this regard, our work suggests that when the classical post-processing units at the key distillation layer are untrusted, they should not output the final key k_A and k_B but they should output shares of it to the key management layer.[58,59] There, k_A and k_B could be either reconstructed by say Alice and Bob in secure labs, or their shares could be stored in distributed memories for later use, or they could be employed for instance for encryption purposes via say the one-time pad. Importantly, however, all the key generation process at the key distillation layer can be performed with corrupted devices. Also, we note that, if necessary, operations like storage or encryption at the key management layer could also be performed in the presence of corrupted devices by using techniques from secure multiparty computation.[38] The actual management and storage of the shares of k_A and k_B generated by the key distillation layer is responsibility of the key management layer and depends on the particular application. We remark, however, that such layer structure is introduced here mainly for illustrative purposes, as in this paper we consider the key distillation problem, which is the task of the key distillation layer. Our results are rather general and similar techniques could be applied as well to other applications without the need of any particular layer structure.

Before we go to the specifics of our result. We remark that our idea is general and does *not* depend on the specific verifiable SS scheme used. This is an important strength of our idea. Let us start by providing an overview of the general strategy that we follow to achieve our goal, which uses as main ingredients VSS schemes[41–43] and privacy amplification techniques[63] (see Methods section). The former is employed to defeat corrupted classical post-processing units. Indeed, given that $t' < s/3$ and $t'' < r/3$, the use of VSS schemes allows to post-process the raw keys generated by the QKD modules in a distributed setting by acting only on raw key shares. Obviously, if a pair of QKD modules is corrupted, the generated raw key shares could be known to Eve. Importantly, however, we can prove security also in this scenario mainly because VSS does *not* require the dealer to be honest, and the use of privacy amplification techniques can remove any information about the final key which could be known to the corrupted pairs of QKD modules (as it is explained in more detail below). In addition, we remark that the post-processing of raw key shares can be performed such that no set of corrupted classical post-processing units can reconstruct k_A and k_B. Also, VSS guarantees that k_A and k_B is a correct key independently of the misbehaviour of the corrupted QKD modules and classical post-processing units which might wish to purposely introduce errors. Another key insight of our paper is to show that, since all the classical post-processing techniques that are typically applied in QKD are "linear" in nature (i.e., they involve simple functions in linear algebra, such as bit-wise XOR and multiplications of matrices), they are easily implementable in a distributed setting.

Let us illustrate this last point with a simple example. In particular, let us consider, for instance, the error correction step in QKD. Here, say Bob wants to correct a certain bit string, $k_{B,\text{key}}$, to match that of Alice, which we shall denote by $k_{A,\text{key}}$. In general, this process requires that both Alice and Bob first apply certain error correction matrices, M_{EC}, to $k_{A,\text{key}}$ and $k_{B,\text{key}}$ to obtain the syndrome information $s_A = M_{EC}k_{A,\text{key}}$ and $s_B = M_{EC}k_{B,\text{key}}$, respectively. Afterward, if $s_A \neq s_B$ Bob modifies $k_{B,\text{key}}$ accordingly. This process might be repeated a few times until it is guaranteed that $k_{B,\text{key}} = k_{A,\text{key}}$ with high probability. Let us now consider again the same procedure but now acting on shares, $k_{Aj,\text{key}}$ and $k_{Bj,\text{key}}$, of $k_{A,\text{key}}$ and $k_{B,\text{key}}$, respectively. That is, say $k_{A,\text{key}} = \oplus_j^q k_{Aj,\text{key}}$ and $k_{B,\text{key}} = \oplus_j^q k_{Bj,\text{key}}$, with q being the total number of shares. For this, Alice and Bob first apply M_{EC} to $k_{Aj,\text{key}}$ and $k_{Bj,\text{key}}$ to obtain $s_{Aj} = M_{EC}k_{Aj,\text{key}}$ and $s_{Bj} = M_{EC}k_{Bj,\text{key}}$, respectively, for all j. Next, Alice sends s_{Aj} to Bob who obtains $s_A = \oplus_{j=1}^q s_{Aj}$ and $s_B = \oplus_{j=1}^q s_{Bj}$. This is so because $\oplus_{j=1}^q s_{Aj} = \oplus_{j=1}^q M_{EC}k_{Aj,\text{key}} = M_{EC} \oplus_{j=1}^q k_{Aj,\text{key}} = M_{EC}k_{A,\text{key}} = s_A$, and a similar argument applies to s_B. Finally, if $s_A \neq s_B$ Bob corrects $k_{B,\text{key}}$ by acting on its shares $k_{Bj,\text{key}}$. Note that to flip certain bits in $k_{B,\text{key}}$ is equivalent to flip the corresponding bits in one of its shares $k_{Bj,\text{key}}$. That is, error correction in QKD can be easily performed in a distributed setting by acting only on shares of $k_{A,\text{key}}$ and $k_{B,\text{key}}$. The same argument applies as well to the other classical post-processing techniques in QKD, as all of them involve only linear operations.

To defeat corrupted QKD modules, on the other hand, we use privacy amplification techniques. Suppose, for instance, that each pair QKD$_{Ai}$ and QKD$_{Bi}$ outputs a raw key, k'_{Ai} and k'_{Bi}. Moreover, suppose for the moment that the classical post-processing units are trusted and they distil a supposedly $(\varepsilon_{\text{cor}}/n)$-correct and $(\varepsilon_{\text{sec}}/n)$-secret key, k''_{Ai} and k''_{Bi}, of length N bits from each pair k'_{Ai} and k'_{Bi}. Then, the $n \times N$ bit strings $k'_A = [k''_{A1}, \ldots, k''_{An}]$ and $k'_B = [k''_{B1}, \ldots, k''_{Bn}]$ are for certain ε_{cor}-correct. The secrecy condition, however, only holds if all the QKD modules are trusted. If say the pair QKD$_{Ai'}$ and QKD$_{Bi'}$ is corrupted then the key strings $k''_{Ai'}$ and $k''_{Bi'}$ are compromised. So, given that $t < n$ the classical post-processing units can apply privacy amplification to k'_A and k'_B to extract two shorter $(n - t) \times N$ bit strings, k_A and k_B, which are ε_{sec}-secret and thus ε-secure. In the presence of untrusted classical post-processing units, this process can be performed in a distributed manner by acting on data shares.

In short, the general strategy can be decomposed in three main steps, which are illustrated in Fig. 4b. First, each pair of QKD modules generates a raw key and the protocol information and sends them to the CP units. Second, the CP units distil a supposedly $(\varepsilon_{\text{cor}}/n)$-correct and $(\varepsilon_{\text{sec}}/n)$-secret key from each raw key received and concatenate the resulting keys to form a longer key bit string, which, in the absence of malicious QKD modules, would be then ε-secure. Finally, in the third step, the CP units apply privacy amplification to this longer bit string to remove any information that could be known to Eve due to the presence of malicious QKD modules. If the CP units are untrusted, these steps are performed in a distributed setting by acting on data shares produced by a VSS scheme.

Next we evaluate three different scenarios of practical interest in this context. For concreteness, in these examples we use the VSS scheme introduced in ref. [41] and which is described in the Methods section.

QKD with malicious QKD modules

We begin by analysing the situation where Alice and Bob have n pairs of QKD modules and up to $t < n$ of them could be corrupted, and each of Alice and Bob has one honest classical post-processing unit. This scenario is illustrated in Fig. 5 and corresponds to the case $s = r = 1$ and $t' = t'' = 0$ in Fig. 4a.

A possible solution to this scenario is rather simple; see *Protocol* 1 below. This protocol follows the spirit of "countermeasure 3" in ref. [25] However, in contrast to ref.,[25] which is restricted to measurement devices in a DI-QKD setting, *Protocol* 1 applies to both DI-QKD and non-DI-QKD, and considers the whole QKD key generation devices, which include the sources.

Protocol 1:

1. *Generation of raw keys and protocol information*: Each pair QKD$_{Ai}$ and QKD$_{Bi}$ outputs, respectively, the bit strings k'_{Ai} and $p_{Ai,\text{info}}$, and k'_{Bi} and $p_{Bi,\text{info}}$, or the abort symbol $\perp_i$, $\forall i = 1, \ldots, n$.

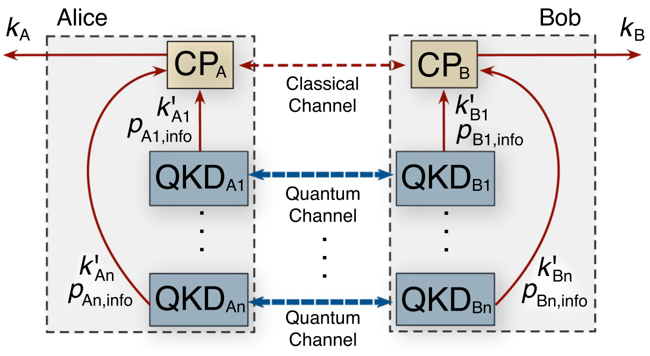

Fig. 5 QKD with malicious QKD modules. Alice and Bob have n pairs of QKD modules, and up to $t < n$ of them could be corrupted. Alice's (Bob's) ith QKD module is supposed to generate a raw key k'_{Ai} (k'_{Bi}) and the protocol information $p_{Ai,\text{info}}$ $(p_{Bi,\text{info}})$, with $i = 1,\ldots,n$. Also, they have one classical post-processing unit each, which is assumed to be honest. The goal is to distil an ε-secure key, k_A and k_B. This can be achieved by using *Protocol* 1. See the main text for further details

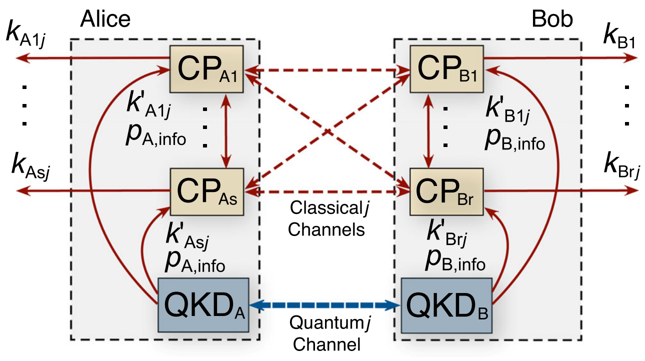

Fig. 6 QKD with malicious classical post-processing units. Alice and Bob have one trusted QKD module each, QKD_A and QKD_B, which generate, respectively, the raw key k'_A and k'_B and the protocol information $p_{A,info}$ and $p_{B,info}$. Also, Alice (Bob) has s (r) classical post-processing units CP_{Ai} ($CP_{Bi'}$) with $i = 1,\ldots,s$ ($i' = 1,\ldots,r$), and up to $t' < s/3$ ($t'' < r/3$) of these units could be corrupted. The goal is to produce shares of an ε-secure key, k_A and k_B, from which the final key could be reconstructed. This can be achieved by using *Protocol* 2. See the Methods section for a detailed description of this protocol. In the figure, k'_{Aij} ($k'_{Bi'j}$) denotes the jth share of k'_A (k'_B) which QKD_A (QKD_B) sends to CP_{Ai} ($CP_{Bi'}$), and k_{Aij} ($k_{Bi'j}$) identifies the jth share of k_A (k_B) that is produced by CP_{Ai} ($CP_{Bi'}$). Since QKD_A (QKD_B) is honest, note that the shares k'_{Aij} ($k'_{Bi'j}$) are equal for all i (i')

2. *Generation of an ε_{cor}-correct key*: The units CP_A and CP_B use the key distillation procedure prescribed by the QKD protocol to generate an (ε_{cor}/n)-correct and (ε_{sec}/n)-secret key, k''_{Ai} and k''_{Bi}, from each pair k'_{Ai} and k'_{Bi}, or they generate the abort symbol $\perp_i$, $\forall i = 1,\ldots,n$. Afterward, CP_A (CP_B) concatenates the $M \le n$ keys k''_{Ai} (k''_{Bi}) which are different from $\perp_i$ to form the bit string $k'_A = [k''_{A1}, \ldots, k''_{AM}]$ ($k'_B = [k''_{B1}, \ldots, k''_{BM}]$). Since by assumption CP_A and CP_B are trusted, k'_A and k'_B are for certain ε_{cor}-correct. The secrecy condition only holds if all k''_{Ai} and k''_{Bi} originate from raw keys output by honest QKD modules. For simplicity, let the length of k''_{Ai} and k''_{Bi} be N bits $\forall i$.
3. *Generation of an ε-secure key*: CP_A and CP_B apply a randomly selected universal$_2$ hash function to k'_A and k'_B to extract two bit strings, k_A and k_B, of length $(M - t) \times N$ bits. k_A and k_B are by definition ε_{sec}-secret, and thus, from step 2, they are ε-secure.

Note that in step 3 of *Protocol* 1 we consider the worst-case scenario where all k''_{Ai} and k''_{Bi} generated by corrupted QKD modules contribute to k'_A and k'_B, respectively, as Alice and Bob cannot discard this case. Most importantly, *Protocol* 1 allows Alice and Bob to defeat *any* covert channel (including, e.g., memory attacks) from the QKD modules, as this protocol guarantees that none of the malicious QKD modules can access k_A or k_B. Our results are summarised in the following Claim, whose proof is directly from the definition of *Protocol* 1.

Claim 1.
Suppose that Alice and Bob have n pairs of QKD modules and up to $t < n$ of them could be corrupted. Also, suppose that they have one trusted classical post-processing unit each. Let $M \le n$ denote the number of pairs of QKD modules that do not abort and whose raw key could in principle be transformed into an (ε/n)-secure key, and let N bits be the length of such supposedly secure key. Protocol 1 allows Alice and Bob to distil an ε-secure key of length $(M - t) \times N$ bits. Moreover, the re-use of the devices does not compromise the security of the keys distilled in previous QKD runs.

Importantly, we remark that *Protocol* 1 is basically optimal with respect to the resulting secret key rate, in the sense that there is no protocol which can deliver a higher key rate in the scenario considered. This is so because of the following. If no pair of QKD modules aborts and its raw data could in principle be transformed into a secure key we have, by definition, that the maximum *total* final key length is at most $n \times N$ bits. Also, we know that up to $t \times N$ bits of such key could be compromised by the t pairs of corrupted QKD modules. That is, the maximum secure key length is at most $(n - t) \times N$ bits. Moreover, as discussed above, if some pairs of QKD modules abort we must necessarily assume the worst-case scenario where they are honest. This is so because through her interaction with the quantum signals in the channel, Eve could always force honest QKD modules to abort by simply increasing the resulting QBER or phase error rate. That is, in this scenario it is not possible to distil a key length greater than $(M - t) \times N$ bits.

QKD with malicious classical post-processing units

We now consider the situation where Alice and Bob have one trusted QKD module each, and Alice (Bob) has s (r) classical post-processing units CP_{Ai} ($CP_{Bi'}$), with $i = 1,\ldots,s$ ($i' = 1,\ldots,r$), and up to $t' < s/3$ ($t'' < r/3$) of them could be corrupted. This is illustrated in Fig. 6 and corresponds to the case $n = 1$ and $t = 0$ in Fig. 4a.

Since now the units CP_{Ai} and $CP_{Bi'}$ could be malicious, we aim to generate shares of an ε-secure key, k_A and k_B. A possible solution to this scenario is given by *Protocol* 2, which is described in detail in the Methods section. The main result is summarised in the following Claim, whose proof is directly from the description of *Protocol* 2:

Claim 2.
Suppose that Alice and Bob have one trusted QKD module each, and each of them has, respectively, s and r classical post-processing units. Also, suppose that up to $t' < s/3$ of Alice's units and up to $t'' < r/3$ of Bob's units could be corrupted. Then, if we disregard the cost of authenticating the classical channels between Alice and Bob's classical post-processing units, Protocol 2 allows them to distil an ε-secure key of the same length as would be possible in a completely trusted scenario. Moreover, the re-use of the devices does not compromise the security of the keys distilled in previous QKD runs.

That is, if we ignore the cost of authenticating the classical channels between the units CP_{Ai} and $CP_{Bi'}$, Claim 2 implies that *Protocol* 2 is optimal with respect to the resulting secret key length, in the sense that there is no protocol which can deliver a higher key rate in the scenario considered. This is so because this protocol allows Alice and Bob to obtain a secret key of the same length as it would be possible if all devices are trusted. We refer the reader to the Supplementary Note 2 for a simpler but less efficient protocol to achieve the same task.

Also, we remark that in *Protocol* 2 the cost due to authenticating the classical channels is relatively small even when s and r are reasonable large. Indeed, if we assume for simplicity the symmetric scenario where $s = r$ and $t' = t''$ (i.e., each of Alice and Bob has s CP units and up to $t' < s/3$ of them are malicious), it can be shown that this authentication cost is basically $s(2t' + 1)k_{au}$ bits, where k_{au} denotes the authentication cost in the trusted scenario (i.e., when $s = 1$ and $t' = 0$). That is, the overall cost of authentication is only $s(2t' + 1)$ times that of a standard QKD scheme with honest devices. This is so because in the untrusted scenario, for any message that has to be authenticated (say for instance from Alice), it is enough that $2t' + 1$ units CP_{Ai} send that message authenticated to all the s $CP_{Bi'}$ units at Bob's side. In so doing, it is guaranteed that Bob's units can reconstruct the message correctly by using say majority voting. Importantly, the parameter k_{au} is logarithmic in the size of the messages sent, which assures that the cost of authentication in *Protocol* 2 is small.

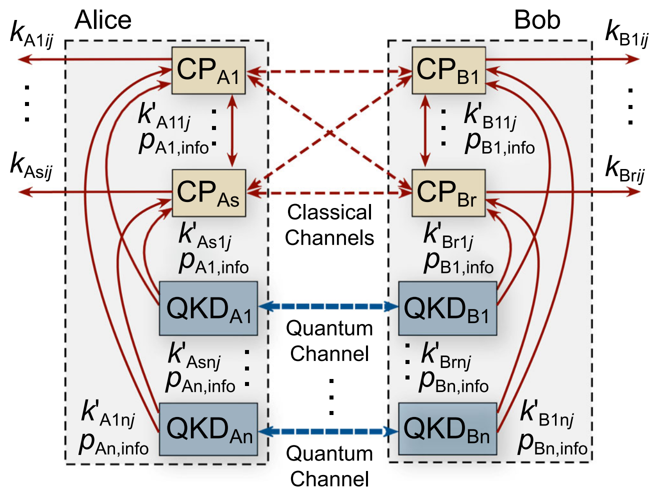

Fig. 7 QKD with malicious QKD modules and classical post-processing units. Alice and Bob have n pairs of QKD modules, QKD_{Ai} and QKD_{Bi}, and up to $t < n$ of them could be corrupted. They generate, respectively, the raw key k'_{Ai} and k'_{Bi} and the protocol information $p_{Ai,info}$ and $p_{Bi,info}$. Also, Alice (Bob) has s (r) classical post-processing units $CP_{Ai'}$ ($CP_{Bi''}$) with $i' = 1,\ldots,s$ ($i'' = 1,\ldots,r$), and up to $t' < s/3$ ($t'' < r/3$) of these units could be corrupted. The goal is to produce shares of an ε-secure key, k_A and k_B, from which the final key could be reconstructed. This can be achieved by using *Protocol* 3. See the main text for further details. In the figure, $k'_{Ai'ij}$ ($k'_{Bi''ij}$) denotes the jth share of k'_{Ai} (k'_{Bi}) which QKD_{Ai} (QKD_{Bi}) sends to $CP_{Ai'}$ ($CP_{Bi''}$), and $k_{Ai'ij}$ ($k_{Bi''ij}$) identifies the shares of k_A (k_B) that are produced by $CP_{Ai'}$ ($CP_{Bi''}$)

QKD with malicious QKD modules and classical post-processing units

Finally, here we consider the situation where Alice and Bob have n pairs of QKD modules, QKD_{Ai} and QKD_{Bi} with $i = 1,\ldots,n$, and Alice (Bob) has s (r) classical post-processing units $CP_{Ai'}$ ($CP_{Bi''}$), with $i' = 1,\ldots,s$ ($i'' = 1,\ldots,r$), and up to $t < n$ pairs of QKD modules, up to $t' < s/3$ units $CP_{Ai'}$ and up to $t'' < r/3$ units $CP_{Bi''}$ could be corrupted. This scenario is illustrated in Fig. 7 and corresponds to the most general case considered in Fig. 4a.

For illustrative purposes, let us discuss first a naive protocol that fails to achieve the goal. In particular, suppose for simplicity that $s = r = n$, and, moreover, we have that up to $t < n$ groups of devices $G_i \equiv \{QKD_{Ai}, QKD_{Bi}, CP_{Ai}, CP_{Bi}\}$ could be corrupted, where we say that a group G_i is corrupted if at least one of its elements is corrupted. Then, if one disregards efficiency issues, a straightforward solution to this scenario might appear to be as follows. Each G_i generates a supposedly ε-secure key, k_{Ai} and k_{Bi}, and then this key is simply considered as the ith share of the final key, k_A and k_B. That is, $k_A = \oplus_{i=1}^{n} k_{Ai}$ and $k_B = \oplus_{i=1}^{n} k_{Bi}$. Indeed, given that $t < n$, k_A and k_B is for certain ε_{sec}-secret. However, the main problem of this naive approach is that k_A and k_B might not be *correct* because a corrupted G_i could output $k_{Ai} \neq k_{Bi}$ and thus $k_A \neq k_B$.

Below we provide a simple solution (*Protocol* 3) to the general scenario. It builds on *Protocols* 1 and 2, and it consists of three main steps.

Protocol 3:

1. *Generation and distribution of shares of* (ε/n)*-secure keys*: Each pair QKD_{Ai} and QKD_{Bi} uses say *Protocol* 2 to distribute shares of an (ε/n)-secure key, k_{Ai} and k_{Bi}, or the abort symbol $\perp_i$, between $CP_{Ai'}$ and $CP_{Bi''}$, respectively. Let $\tilde{k}_{Ai'ij}$ ($\tilde{k}_{Bi''ij'}$) be the jth (j'th) share of k_{Ai} (k_{Bi}) obtained by $CP_{Ai'}$ ($CP_{Bi''}$). For simplicity, we will suppose that the length of k_{Ai} and k_{Bi} is N bits $\forall i$.
2. *Generation of shares of an* ε_{cor}*-correct key*: Let $\vec{0}$ be the N-bit zero vector, and M be the number of k_{Ai} and k_{Bi} which are different from $\perp_i$. Each $CP_{Ai'}$ defines $k''_{Ai'ij} = \left[\vec{0}_1, \ldots, \vec{0}_{i-1}, \tilde{k}_{Ai'ij}, \vec{0}_{i+1}, \ldots, \vec{0}_M\right]$. Likewise, the $CP_{Bi''}$ form $k''_{Bi''ij'}$ from $\tilde{k}_{Bi''ij'}$. $k''_{Ai'ij}$ and $k''_{Bi''ij'}$ are by definition shares of an ε_{cor}-correct key. The secrecy condition only holds if all k_{Ai} and k_{Bi} originate from honest QKD modules.
3. *Generation of shares of an* ε*-secure key*: The $CP_{Ai'}$ use the RBS scheme (see Methods section) to randomly select a universal$_2$ hash function, h_P. Next, they obtain shares, $k_{Ai'ij} = h_P(k''_{Ai'ij})$ of length $(M - t) \times N$ bits of a final key k_A, and say the first $2t' + 1$ $CP_{Ai'}$ send h_P to all $CP_{Bi''}$, which use majority voting to determine h_P from the information received, and then obtain shares $k_{Bi''ij'} = h_P(k''_{Bi''ij'})$ of the final key k_B.

Indeed, given that $t' < M_{Ai}/3$ and $t'' < M_{Bi}/3$ for all $i = 1,\ldots,M$, where M_{Ai} (M_{Bi}) denotes the number of $CP_{Ai'}$ ($CP_{Bi''}$) that do not produce $\perp_i$ but output post-processed shares, $k_{Ai'ij}$ ($k_{Bi''ij'}$), from k_{Ai} (k_{Bi}), then the final key, k_A and k_B, is ε-secure. Also, Alice (Bob) could obtain k_A (k_B) by using the reconstruct protocol of a VSS (see Methods section). That is, Alice (Bob) could use majority voting to obtain the shares k_{Aij} and $k_{Bij'}$ of k_A (k_B) from $k_{Ai'ij}$ ($k_{Bi''ij'}$) for all $i = 1, \ldots, M$ and $j = 1,\ldots,q$ ($j' = 1,\ldots,q'$), and she (he) calculates $k_A = \oplus_{i=1}^{M} \oplus_{j=1}^{q} k_{Aij}$ ($k_B = \oplus_{i=1}^{M} \oplus_{j'=1}^{q'} k_{Bij'}$) where q (q') is the total number of shares of k_{Ai} (k_{Bi}) for each i.

Our results are summarised in the following Claim, whose proof is directly from the definition of *Protocol* 3:

Claim 3.

Suppose that Alice and Bob have n pairs of QKD modules and Alice (Bob) has s (r) classical post-processing units. Suppose that up to $t < n$ *pairs of QKD modules, up to* $t' < s/3$ *classical post-processing units of Alice, and up to* $t'' < r/3$ *classical post-processing units of Bob could be corrupted. Let* $M \leq n$ *denote the number of pairs of QKD modules that do not abort and whose raw key can be transformed into a supposedly* (ε/n)*-secure key, and let N bits be the length of such key. Then Protocol 3 allows Alice and Bob to distil an* ε*-secure key of length* $(M - t) \times N$ *bits. Moreover, the re-use of the devices does not compromise the security of the keys distilled in previous QKD runs.*

We remark that if we disregard the cost of authenticating the classical channels between Alice and Bob's classical post-processing units, *Protocol* 3 is optimal with respect to the resulting secret key length, in the sense that no other protocol can deliver a higher key rate for the scenario considered. The argument follows directly from that used in the subsection above that evaluates QKD with malicious QKD modules, where we showed that if the classical post-processing units are trusted the secret key rate is upper bounded by $(M - t) \times N$ bits. So, in the presence of corrupted classical post-processing units this upper bound also trivially holds.

Moreover, we note that, like in the case of *Protocol* 2, the cost of authenticating the classical channels between the units $CP_{Ai'}$ and $CP_{Bi''}$ in *Protocol* 3 is also relatively small even when n, s and r are fairly large. Indeed, if we assume again the symmetric scenario where $s = r$ and $t' = t''$, it can be shown that this authentication cost is now roughly $\approx s(2t' + 1)(nk_{au} + k'_{pa})$ bits, where k_{au} denotes again the authentication cost in the trusted scenario (i.e., when $n = s = 1$ and $t = t' = 0$). The term $ns(2t' + 1)k_{au}$ comes mainly because here we run n times *Protocol* 2, each time for a different pair of modules QKD_{Ai} and QKD_{Bi}. The term $s(2t' + 1)k'_{pa}$, on the other hand, considers the authentication cost of step 3 in *Protocol* 3. This step requires that $2t' + 1$ units $CP_{Ai'}$ send the function h_P to all the s $CP_{Bi'''}$, being the term k'_{pa} logarithmic in the size of the message sent. That is, the overall cost of authentication of *Protocol* 3 is only about $\approx ns(2t' + 1)$ times that of a standard

QKD scheme with honest devices, which demonstrates the practical feasibility of our approach.

DISCUSSION

Security proofs of QKD typically assume that all devices are honest and there are no covert channels that leak information to Eve. Unfortunately, however, these assumptions are very hard, if not impossible, to guarantee in practice. Indeed, our work highlights the folly of bringing an untrusted device into a trusted lab, which has been known since the time of the Trojans.

Memory attacks[25] constitute an example of how covert channels can severely jeopardise the security of both DI-QKD and non-DI-QKD. These attacks demonstrate that quantum mechanics alone is not enough to guarantee the security of practical QKD realisations but, for this, one needs to resort to additional assumptions. Also, recent results on Trojan Horse attacks[31–36,69,70] against conventional cryptography underline the vulnerabilities of the classical post-processing units in QKD, and this threat is expected to only rise with time.

In this paper, we have introduced a simple method to reduce the risk due to this type of attacks; it is based on the idea of classical error correction via redundancy. We use the assumption that, for Eve, it might be more difficult to tamper with several devices from different vendors than corrupting just one QKD device from a single vendor, which indeed is probably the best Alice and Bob can hope for. With this assumption in place, and given that there is at least one pair of honest QKD modules and that the number of corrupted classical post-processing units is less than one-third of them, we have shown how VSS schemes, together with privacy amplification techniques, could be used to reliable distribute a secret key in the presence of malicious devices and thus re-establish the security of QKD.

VSS and SS techniques have been used previously in quantum information.[71–73] For instance, the authors of refs. [71,72] proposed a quantum version of VSS to achieve secure multiparty quantum computation, while in ref. [73] classical SS schemes are combined with QKD to achieve information-theoretically secure distributed storage systems.

A key insight of our paper is very simple yet potentially very useful: the typical classical post-processing in QKD involves only operations which are "linear" in nature, and thus they could be easily implemented in a distributed setting by acting on data shares from say a linear VSS scheme. Also, our results suggest that the way QKD is sometimes envisioned commercially (e.g., as a single "box" which directly outputs the final keys),[74] should perhaps be replaced with a "modular design", where one divides up a QKD system into various modules (e.g., purely quantum modules and classical post-processing units) and use them to cross-check each other for security, which can be done in an independent manner as we have shown above.

To illustrate our results, we have proposed specific protocols for three scenarios of practical interest. They assume that either the QKD modules, the classical post-processing units, or both of them together could be corrupted. Remarkably, if we disregard the cost of classical authentication, all these protocols are optimal with respect to the secret key rate. They use the VSS scheme introduced in ref. [41] which is very simple to implement. Its main drawback is, however, that, for a given number of corrupted parties, the number of shares grows exponentially with the total number of parties. Nevertheless, for a small number of parties (which is the scenario that we are interested in QKD), the protocol is efficient in terms of computational complexity. Also, we remark that there exist efficient three-round VSS protocols where the computation and communication required is polynomial in the total number of parties.[75] Moreover, these schemes use a minimum number of communication rounds,[76] and they could also be used here. It would be interesting to further investigate the most resource efficient protocols to be used in the QKD framework.

Finally, we would like to emphasise that our approach does not jeopardise the potential practical advantages of QKD over classical competitors in any way. Indeed, as we have already mentioned previously, covert channels constitute a fundamental problem for conventional cryptography as well. To foil covert channels is therefore equally essential for both frameworks. This means that redundancy will be needed in both scenarios and, thus, our work does not shift the balance of power between them in any sense.

METHODS

Secure multiparty computation toolbox

Here we briefly introduce some definitions and cryptographic protocols that are used in the main text; they are mainly taken from refs. [38,41]

We consider a scenario with a dealer and n parties, and we suppose a threshold active adversary structure where Eve can actively corrupt the dealer and up to t of the parties. Active corruption means that Eve can fully control the corrupted parties whose behaviour can deviate arbitrarily from the protocol's prescriptions. We refer the reader to the Supplementary Note 3 for the modelling of more general adversary structures. For simplicity, we assume that all messages are binary strings and the symbol $\oplus$ below denotes bit-wise XOR or bit-wise addition modulo 2. We remark, however, that the the protocols below work as well over any finite field or ring.

In this scenario, a n-out-of-q threshold SS scheme[77,78] is a protocol that allows the dealer to split a message m between n parties such that, if he is honest, any group of q or more parties can collaborate to reconstruct m but no group with less than q parties can obtain any information about m. If $n=q$, this could be achieved by splitting m into a random sum of q shares m_i. That is, one selects the first $q-1$ shares m_i of m at random, and then chooses $m_q=m\oplus m_1\oplus\ldots\oplus m_{q-1}$.[38]

A drawback of SS schemes is that they do not guarantee the consistency of the shares, which is essential to assure the correctness of the keys delivered by the QKD protocols in the main text. That is, during the reconstruct phase of a SS scheme, corrupted parties could send different m_i to the honest parties such that they obtain different values for m. This problem can be solved with VSS schemes,[42,43] which distribute m_i in a redundant manner such that the honest parties can use error correction to obtain the correct values. Indeed, provided that the necessary and sufficient conditon $t<n/3$ is satisfied, a VSS scheme guarantees that there exists a well-defined m that all honest parties obtain from their shares.[39–41]

The share and reconstruct protocols of a VSS scheme satisfy three conditions. First, independently of whether or not the dealer is honest, if the share protocol is successful then the reconstruct protocol delivers the same m to all the honest parties. Second, if the dealer is honest, the value of the reconstructed m coincides with that provided by the dealer. And third, if the dealer is honest, the information obtained by any set of t or less parties after the share (reconstruct) protocol is independent of any previous information that they held before the protocol (is just the reconstructed bit string m). Below we present a simple VSS scheme that builds on the q-out-of-q threshold SS protocol above,[41] and which we use in *Protocols* 2 and 3. Importantly, given that $t<n/3$, this scheme provides information-theoretic security.[41] See the Supplementary Note 1 and the Supplementary Figure 1 for a graphical representation of its share and reconstruct protocols.

Share protocol:

1. The dealer uses a q-out-of-q SS scheme to split m into $q=\binom{n}{n-t}$ shares m_i, with $i=1,\ldots,q$.
2. Let $\{\sigma_1,\ldots,\sigma_q\}$ denote all $(n-t)$-combinations of the set of n parties. Then, for each $i=1,\ldots,q$, the dealer sends m_i over a secure channel (i.e., a channel that provides secrecy and authentication) to each party in σ_i. If a party does not receive his share, he takes as default share say a zero bit string.
3. All pairs of parties in σ_i send each other their shares m_i over a secure channel to check if they are indeed equal. If an inconsistency is found, they complain using a broadcast channel.
4. If a complaint is raised in σ_i, the dealer broadcasts m_i to all parties and they accept the share received. Otherwise, the protocol aborts.

Reconstruct protocol:

1. All pairs of parties send each other their shares over an authenticated channel.
2. Each party uses majority voting to reconstruct the shares m_i $\forall i$, and then obtains $m = \oplus_{i=1}^{q} m_i$.

From the description above, it is guaranteed that when the share protocol is successful (i.e., it does not abort), all the honest parties who received the ith share of m obtain exactly the same bit string m_i. Also, this protocol assures that any share m_i of m is distributed to at least $2t+1$ different parties. This is so because this is the minimum size of any set σ_i. This means, in particular, that, since the number of corrupted parties is at most t, the use of a decision rule based on majority voting in the reconstruct protocol permits all the honest parties to obtain the same fixed m_i for all i. Moreover, it is straightforward to show that when the dealer is honest, the reconstructed message m is equal to his original message. Furthermore, we have that m is only revealed to the parties once the reconstruct phase ends. This is so because at least one bit string m_i is only shared by honest parties since there is at least one set σ_i which does not contain any corrupted party. Also, note that if a complaint is raised in a certain σ_i during the share protocol, the fact that the dealer broadcast m_i to all parties does not violate secrecy. This is so because a complaint can only occur if either the dealer is corrupted or σ_i contains at least one corrupted player, hence the adversary knew m_i already.

We remark that the broadcast channel which is required in steps 3 and 4 of the share protocol can be a simulated channel. Indeed, given that $t < n/3$, there exist efficient poly(n) protocols that can simulate a broadcast channel with information-theoretic security in an optimal number of $t+1$ communication rounds.[79,80] Furthermore, if a physical broadcast channel is actually available, there exist efficient information-theoretically secure VSS schemes that only require a majority of honest parties (i.e., $t < n/2$) and which could also be used in this context.[43]

Next we present a simple scheme to generate a common perfectly unbiased random l-bit string (RBS) r between n parties when up to $t < n/3$ of them could be corrupted. It follows directly from VSS.[39–41] For convenience, we call it the RBS protocol. We use it to randomly select universal$_2$ hash functions in *Protocols* 2 and 3 in the main text, where we cannot assume the existence of an external honest dealer which provides them to the QKD devices. The RBS scheme allows mutually untrusted parties to generate and share random numbers through discussions.

RBS protocol:

1. Say each of the first $t+1$ parties produces locally a random l-bit string r_i and sends it to all the other parties using the share protocol above.
2. Each party uses a broadcast channel to confirm that they have received all their shares from the first $t+1$ parties. Otherwise, the protocol aborts.
3. All parties use the reconstruct protocol above to obtain r_i for all $i = 1,\ldots,t+1$. Afterward, each of them calculates locally $r = \oplus_{i=1}^{t+1} r_i$.

It is straightforward to show that this protocol guarantees that all honest parties share a perfectly unbiased random bit string r. The use of the share and the reconstruct protocols of a VSS scheme assures that all honest parties reconstruct the same r_i $\forall i$ and thus the same r. In addition, step 2 of the protocol guarantees that the first $t+1$ parties generate and distribute their strings r_i before knowing the strings of the other parties. Moreover, since the number of corrupted parties is at most t, we have that at least one honest party generates a truly random bit string r_i, and thus r is also random.

Protocol 2

Here we present the different steps of *Protocol* 2 in detail. For concreteness, whenever we refer to the share and reconstruct protocols of a VSS scheme we mean those presented in the previous section, which have been introduced in ref. [41] but in principle any VSS scheme could be used.

Also, to simplify the discussion, in *Protocol* 2 we consider the case where $p_{\rm A,info}$ and $p_{\rm B,info}$ determine the sifting procedure of the QKD scheme in a deterministic way. That is, there is no *random* post-selection of data from the raw key. In addition, we assume that Alice and Bob do not estimate the actual QBER but they apply error correction for a pre-fixed QBER value followed by an error verification step. However, we remark that *Protocol* 2 could be straightforwardly adapted to cover also these two scenarios.

1. *Generation and distribution of shares of raw keys and protocol information*: QKD$_{\rm A}$ and QKD$_{\rm B}$ obtain, respectively, the raw keys $k_{\rm A}'$ and $k_{\rm B}'$ and the protocol information $p_{\rm A,info}$ and $p_{\rm B,info}$, or the abort symbol $\perp$. If the result is different from $\perp$, QKD$_{\rm A}$ uses the share protocol of a VSS scheme to create $q = \binom{s}{s-t'}$ shares of $k_{\rm A}'$ and distributes them among the CP$_{{\rm A}i}$, with $i = 1,\ldots,s$. Likewise, QKD$_{\rm B}$ creates $q' = \binom{r}{r-t''}$ shares of $k_{\rm B}'$ and distributes them among the CP$_{{\rm B}i'}$, with $i' = 1,\ldots,r$. Let $k_{{\rm A}ij}'$ ($k_{{\rm B}i'j'}'$) be the jth (j'th) share of $k_{\rm A}'$ $\left(k_{\rm B}'\right)$ received by CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$), with $j = 1,\ldots,q$ ($j' = 1,\ldots,q'$). Also, QKD$_{\rm A}$ (QKD$_{\rm B}$) sends $p_{\rm A,info}$ ($p_{\rm B,info}$) to all CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$). Since by assumption QKD$_{\rm A}$ (QKD$_{\rm B}$) is honest, all CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) receive the same $p_{\rm A,info}$ ($p_{\rm B,info}$) and the shares $k_{{\rm A}ij}'$ $\left(k_{{\rm B}i'j'}'\right)$ are equal for all i (i'). Next, say the first $2t''+1$ CP$_{{\rm B}i'}$ send $p_{\rm B,info}$ to all CP$_{{\rm A}i}$. Likewise, say the first $2t'+1$ CP$_{{\rm A}i}$ send $p_{\rm A,info}$ (for the detected events) to all CP$_{{\rm B}i'}$. Each CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) uses majority voting to determine $p_{\rm B,info}$ ($p_{\rm A,info}$) from the information received. Note that since by assumption the number of corrupted units CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) is at most t' (t''), $2t'+1$ ($2t''+1$) copies of $p_{\rm A,info}$ ($p_{\rm B,info}$) is enough for the honest parties to be able to reconstruct the correct value of these bit strings by using majority voting.
2. *Sifting* Each CP$_{{\rm A}i}$ uses $p_{\rm A,info}$ and $p_{\rm B,info}$ to obtain two bit strings, $k_{{\rm A}ij,{\rm key}}$ and $k_{{\rm A}ij,{\rm est}}$, from $k_{{\rm A}ij}'$. The former (latter) bit string is the part of $k_{{\rm A}ij}'$ that is used for key generation (parameter estimation). Likewise, Bob's CP$_{{\rm B}i'}$ do the same with $k_{{\rm B}i'j'}'$ and obtain $k_{{\rm B}i'j',{\rm key}}$ and $k_{{\rm B}i'j',{\rm est}}$.
3. *Parameter estimation*: All CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) use the reconstruct protocol of a VSS scheme to obtain $k_{\rm A,est}$ ($k_{\rm B,est}$), which is the part of $k_{\rm A}'$ $\left(k_{\rm B}'\right)$ that is used for parameter estimation. For this, they send each other their shares $k_{{\rm A}ij,{\rm est}}$ ($k_{{\rm B}i'j',{\rm est}}$), and each of them uses majority voting to obtain $k_{{\rm A}j,{\rm est}}$ ($k_{{\rm B}j',{\rm est}}$) for all $j = 1,\ldots,q$ ($j' = 1,\ldots,q'$). Afterward, they calculate $k_{{\rm A},est} = \oplus_{j=1}^{q} k_{{\rm A}j,est}$ ($k_{{\rm B},est} = \oplus_{j'=1}^{q'} k_{{\rm B}j',est}$). Next, say the first $2t''+1$ CP$_{{\rm B}i'}$ send $k_{\rm B,est}$ to all CP$_{{\rm A}i}$. Likewise, say the first $2t'+1$ CP$_{{\rm A}i}$ send $k_{\rm A,est}$ to all CP$_{{\rm B}i'}$. Finally, by using majority voting each CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) determines $k_{\rm B,est}$ ($k_{\rm A,est}$). With $p_{\rm A,info}$, $p_{\rm B,info}$, $k_{\rm A,est}$ and $k_{\rm B,est}$, each CP$_{{\rm A}i}$ and CP$_{{\rm B}i'}$ performs locally the parameter estimation step of the protocol (e.g., they estimate the phase error rate). If the estimated values exceed certain tolerated values, they abort.
4. *Error correction*: The CP$_{{\rm A}i}$ and CP$_{{\rm B}i'}$ perform error correction (for a pre-fixed QBER value) on the parts of $k_{\rm A}'$ and $k_{\rm B}'$ that are used for key distillation, which we denote by $k_{\rm A,key}$ and $k_{\rm B,key}$, by acting on their shares $k_{{\rm A}ij,{\rm key}}$ and $k_{{\rm B}i'j',{\rm key}}$ respectively. For this, each CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) applies certain matrices $M_{\rm EC}$ to $k_{{\rm A}ij,{\rm key}}$ ($k_{{\rm B}i'j',{\rm key}}$) to obtain $s_{{\rm A}ij} = M_{\rm EC} k_{{\rm A}ij,{\rm key}}$ ($s_{{\rm B}i'j'} = M_{\rm EC} k_{{\rm B}i'j',{\rm key}}$). Afterward, the CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) use the reconstruct protocol of a VSS scheme to obtain $s_{\rm A} = M_{\rm EC} k_{\rm A,key}$ ($s_{\rm B} = M_{\rm EC} k_{\rm B,key}$). That is, all CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) first send to each other the bit strings $s_{{\rm A}ij}$ ($s_{{\rm B}i'j'}$), and each of them uses majority voting to reconstruct locally $s_{{\rm A}j}$ ($s_{{\rm B}j'}$) $\forall j$ (j') from $s_{{\rm A}ij}$ ($s_{{\rm B}i'j'}$). Then, each CP$_{{\rm A}i}$ (CP$_{{\rm B}i'}$) obtains $s_{\rm A} = \oplus_{j=1}^{q} s_{{\rm A}j}$ $\left(s_{\rm B} = \oplus_{j'=1}^{q'} s_{{\rm B}j'}\right)$. Next, say the first $2t'+1$ CP$_{{\rm A}i}$ send $s_{\rm A}$ to all CP$_{{\rm B}i'}$, and each CP$_{{\rm B}i'}$ uses majority voting to determine $s_{\rm A}$ from the information received. Finally, Bob corrects $k_{\rm B,key}$. For this, say all CP$_{{\rm B}i'}$ which have the j'th share $k_{{\rm B}i'j',{\rm key}}$ for a pre-fixed index $j' = 1,\ldots,q'$, flip certain bits of this share depending on the actual values of $s_{\rm A}$ and $s_{\rm B}$. This whole process is repeated until the error correction procedure ends. Let $\hat{k}_{{\rm A}ij,{\rm key}}$ and $\hat{k}_{{\rm B}i'j',{\rm key}}$ denote the shares $k_{{\rm A}ij,{\rm key}}$ and $k_{{\rm B}i'j',{\rm key}}$ after error correction, and let leak$_{\rm EC}$ bits be the syndrome information interchanged between Alice and Bob during this step. That is, $\hat{k}_{{\rm A}ij,{\rm key}}$ and $\hat{k}_{{\rm B}i'j',{\rm key}}$ are actually equal to $k_{{\rm A}ij,{\rm key}}$ and $k_{{\rm B}i'j',{\rm key}}$ except for the bit strings $k_{{\rm B}i'j',{\rm key}}$ whose bits have been flipped during error correction.
5. *Error verification*: All CP$_{{\rm A}i}$ and CP$_{{\rm B}i'}$ check that the error correction step was indeed successful. For this, the CP$_{{\rm A}i}$ use the RBS scheme introduced in the previous section to randomly select a universal$_2$ hash function, $h_{\rm V}$. Then, they compute a hash $h_{{\rm A}ij} = h_{\rm V}\left(\hat{k}_{{\rm A}ij,{\rm key}}\right)$ of length $\lceil \log_2 (1/\varepsilon_{\rm cor}) \rceil$ bits, and each CP$_{{\rm A}i}$ uses the reconstruct protocol of a VSS scheme to obtain $h_{\rm A} = \oplus_{j=1}^{q} h_{{\rm A}j}$ from $h_{{\rm A}ij}$. That is, they send each other $h_{{\rm A}ij}$ and they use majority voting to determine $h_{{\rm A}j}$ $\forall j$ from $h_{{\rm A}ij}$. Next, say the first $2t'+1$ CP$_{{\rm A}i}$ send $h_{\rm V}$ and $h_{\rm A}$ to all CP$_{{\rm B}i'}$, which determine the correct values of these two quantities by means of majority voting. Also, the units CP$_{{\rm B}i'}$ use the reconstruct protocol of a VSS scheme to obtain $h_{\rm B} = \oplus_{j'=1}^{q'} h_{{\rm B}j'}$. Finally, each unit CP$_{{\rm B}i'}$ checks locally whether or not $h_{\rm A} = h_{\rm B}$. If they are not equal, the

protocol aborts. If they are equal, it is guaranteed that the bit strings $\hat{k}_{\mathrm{A,key}} = \oplus_{j=1}^{q} \hat{k}_{\mathrm{A}j,\mathrm{key}}$ and $\hat{k}_{\mathrm{B,key}} = \oplus_{j'=1}^{q'} \hat{k}_{\mathrm{B}j',\mathrm{key}}$ are equal except for a minuscule probability $\varepsilon_{\mathrm{cor}}$, where $\hat{k}_{\mathrm{A}j,\mathrm{key}}$ ($\hat{k}_{\mathrm{B}j',\mathrm{key}}$) are obtained from $\hat{k}_{\mathrm{A}ij,\mathrm{key}}$ ($\hat{k}_{\mathrm{B}i'j',\mathrm{key}}$) by using majority voting.

6. *Generation of shares of an ε-secure key*: All $\mathrm{CP}_{\mathrm{A}i}$ and $\mathrm{CP}_{\mathrm{B}i'}$ extract from $\hat{k}_{\mathrm{A,key}}$ and $\hat{k}_{\mathrm{B,key}}$ the shares of an $\varepsilon_{\mathrm{sec}}$-secret key, k_{A} and k_{B}. For this, the $\mathrm{CP}_{\mathrm{A}i}$ use the RBS scheme to randomly select a proper universal$_2$ hash function, h_{P}. Next, they obtain $k_{\mathrm{A}ij} = h_{\mathrm{P}}(\hat{k}_{\mathrm{A}ij,\mathrm{key}})$ and say the first $2t'+1$ $\mathrm{CP}_{\mathrm{A}i}$ send h_{P} to all $\mathrm{CP}_{\mathrm{B}i'}$. Bob's CP units use majority voting to determine h_{P} from the information received and they calculate $k_{\mathrm{B}i'j'} = h_{\mathrm{P}}(\hat{k}_{\mathrm{B}i'j',\mathrm{key}})$. The function h_{P} removes Eve's information from $\hat{k}_{\mathrm{A,key}}$, which includes the syndrome information $\mathrm{leak}_{\mathrm{EC}}$ disclosed during error correction, the hash value of length $[\log_2(1/\varepsilon_{\mathrm{cor}})]$ bits disclosed during error verification, and Eve's information about the key according to the estimated phase error rate.

From the description of *Protocol* 2 we have that when $t' < M_{\mathrm{A}}/3$ and $t'' < M_{\mathrm{B}}/3$, where M_{A} (M_{B}) denotes the number of $\mathrm{CP}_{\mathrm{A}i}$ ($\mathrm{CP}_{\mathrm{B}i'}$) that do not abort, then the final key, k_{A} and k_{B}, is ε-secure. This is so because the condition $t' < M_{\mathrm{A}}/3$ (or, equivalently, $s - t' - (s - M_{\mathrm{A}}) > 2t'$) guarantees that for all $j = 1, \ldots, q$, there are at least $2t'+1$ units $\mathrm{CP}_{\mathrm{A}i}$ which send shares $k_{\mathrm{A}ij}$ to Alice. To see this, note that each share $k_{\mathrm{A}j}$, for all j, is held by $s - t'$ units $\mathrm{CP}_{\mathrm{A}i}$, and by assumption we have that at most $s - M_{\mathrm{A}}$ of them could have aborted. A similar argument applies to the condition $t'' < M_{\mathrm{B}}/3$.

DATA AVAILABILITY

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ACKNOWLEDGEMENTS

We thank Charles Bennett and Christian Schaffner for very interesting discussions on the topics of this paper. We acknowledge support from the Galician Regional Government (consolidation of Research Units: AtlantTIC), the Spanish Ministry of Economy and Competitiveness (MINECO), the Fondo Europeo de Desarrollo Regional (FEDER) through Grant Nos. TEC2014-54898-R and TEC2017-88243-R, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 675662 (project QCALL), NSERC, CFI and ORF.

AUTHOR CONTRIBUTIONS

All authors researched, collated, and wrote this paper.

ADDITIONAL INFORMATION

Supplementary Information accompanies the paper on the *npj Quantum Information* website (https://doi.org/10.1038/s41534-019-0131-5).

Competing interests: The authors declare no competing interests.

REFERENCES

1. Gisin, N., Ribordy, R., Tittel, W. & Zbinden, H. Quantum cryptography. *Rev. Mod. Phys.* **74**, 145–195 (2002).
2. Lo, H.-K., Curty, M. & Tamaki, K. Secure quantum key distribution. *Nat. Photon.* **8**, 595–604 (2014).
3. Bennett, C. H. & Brassard, G. Quantum cryptography: public key distribution and coin tossing. In *Proc. IEEE International Conference on Computers, Systems & Signal Processing* 175–179 (IEEE, NY, Bangalore, India, 1984).
4. Ekert, A. K. Quantum cryptography based on Bell's theorem. *Phys. Rev. Lett.* **67**, 661–663 (1991).
5. Peng, C.-Z. et al. Experimental long-distance decoy-state quantum key distribution based on polarization encoding. *Phys. Rev. Lett.* **98**, 010505 (2007).
6. Rosenberg, D. et al. Long-distance decoy-state quantum key distribution in optical fiber. *Phys. Rev. Lett.* **98**, 010503 (2007).
7. Yuan, Z. L., Sharpe, A. W. & Shields, A. J. Unconditionally secure one-way quantum key distribution using decoy pulses. *Appl. Phys. Lett.* **90**, 011118 (2007).
8. Ursin, R. et al. Entanglement-based quantum communication over 144 km. *Nat. Phys.* **3**, 481–486 (2007).
9. Zhao, Y., Fung, C.-H. F., Qi, B., Chen, C. & Lo, H.-K. Quantum hacking: experimental demonstration of time-shift attack against practical quantum-key-distribution systems. *Phys. Rev. A* **78**, 042333 (2008).
10. Nauerth, S., Fürst, M., Schmitt-Manderbach, T., Weier, H. & Weinfurter, H. Information leakage via side channels in freespace BB84 quantum cryptography. *New J. Phys.* **11**, 065001 (2009).
11. Xu, F., Qi, B. & Lo, H.-K. Experimental demonstration of phase-remapping attack in a practical quantum key distribution system. *New J. Phys.* **12**, 113026 (2010).
12. Lydersen, L. et al. Hacking commercial quantum cryptography systems by tailored bright illumination. *Nat. Photon.* **4**, 686–689 (2010).
13. Weier, H. et al. Quantum eavesdropping without interception: an attack exploiting the dead time of single-photon detectors. *New J. Phys.* **13**, 073024 (2011).
14. Mayers, D. & Yao, A. Quantum cryptography with imperfect apparatus. In *Proc. of the 39th Annual Symposium on Foundations of Computer Science (FOCS'98)* 503–509 (IEEE Computer Society, Los Alamitos, California, 1998).
15. Acín, A. et al. Device-independent security of quantum cryptography against collective attacks. *Phys. Rev. Lett.* **98**, 230501 (2007).
16. Vazirani, U. & Vidick, T. Fully device independent quantum key distribution. *Phys. Rev. Lett.* **113**, 140501 (2014).
17. Braunstein, S. L. & Pirandola, S. Side-channel-free quantum key distribution. *Phys. Rev. Lett.* **108**, 130502 (2012).
18. Bell, J. S. On the Einstein–Podolsky–Rosen paradox. *Physics* **1**, 195–200 (1964).
19. Clauser, J. F., Horne, M. A., Shimony, A. & Holt, R. A. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.* **23**, 880–884 (1969).
20. Hensen, B. et al. Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres. *Nature* **526**, 682–686 (2015).
21. Shalm, L. K. et al. A strong loophole-free test of local realism. *Phys. Rev. Lett.* **115**, 250402 (2015).
22. Giustina, M. et al. Significant-loophole-free test of Bell's theorem with entangled photons. *Phys. Rev. Lett.* **115**, 250401 (2015).
23. Hensen, B. et al. Loophole-free Bell test using electron spins in diamond: second experiment and additional analysis. *Sci. Rep.* **6**, 30289 (2016).
24. Rosenfeld, W. et al. Event-ready Bell-test using entangled atoms simultaneously closing detection and locality loopholes. *Phys. Rev. Lett.* **119**, 010402 (2017).
25. Barrett, J., Colbeck, R. & Kent, A. Memory attacks on device-independent quantum cryptography. *Phys. Rev. Lett.* **110**, 010503 (2013).
26. Zander, S., Armitage, G. & Branch, P. A survey of covert channels and countermeasures in computer network protocols. *IEEE Commun. Surv. Tutor. Arch.* **9**, 44–57 (2007).
27. Lampson., B. W. A note on the confinement problem. *Commun. ACM* **16**, 613–615 (1973).
28. United States Government Department of Defense. *Trusted Computer System Evaluation Criteria. Standard 5200.28-STD*. National Computer Security Center. http://csrc.nist.gov/publications/history/dod85.pdf (1985).
29. Gligor, V. D. *A Guide to Understanding Covert Channel Analysis of Trusted Systems (Light Pink Book)*. National Computer Security Center, NCSC-TG-030 edition. http://www.dtic.mil/dtic/tr/fulltext/u2/a276418.pdf (1993).
30. *The Raymond EMC Modular R.F. Shielded Enclosure System*. http://raymondemc.ca/products/products1.htm#5
31. Mitra, S., Wong, H.,-S. P. & Wong, S. *Stopping Hardware Trojans in Their Tracks*. IEEE Spectrum. http://spectrum.ieee.org/semiconductors/design/stopping-hardware-trojans-in-their-tracks (2015)
32. Yang, K., Hicks, M., Dong, Q., Austin, T. & Sylvester, D. A2: analog malicious hardware. In *Proc. IEEE Symposium on Security and Privacy (SP'2016)*, 18–37 (IEEE Computer Society, Los Alamitos, California, 2016).
33. Becker, G. T., Regazzoni, F., Paar, C. & Burleson, W. P. Stealthy Dopant-level Hardware Trojans. *Proc. of the 15th international conference on Cryptographic Hardware and Embedded Systems (CHES'13)* (pp. 197–214. Springer-Verlag Berlin, Heidelberg, 2013).
34. Robertson, J. & Riley, M. *The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies*. Bloomberg LP. http://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies (2018).
35. Prevelakis, V. & Spinellis, D. *The Athens Affair. IEEE Spectrum*. http://spectrum.ieee.org/telecom/security/the-athens-affair (2007).
36. Shrout, R. *Intel Suffers an Epic Security Fail, Offering a Big Opportunity for AMD. MarketWatch: Stock Market News—Financial News*. https://www.marketwatch.com/story/intel-suffers-an-epic-security-fail-offering-a-big-opportunity-for-amd-2018-01-03 (2018).
37. Adee, S. *The Hunt for the Kill Switch. IEEE Spectrum*. https://spectrum.ieee.org/semiconductors/design/the-hunt-for-the-kill-switch (2008).

38. Cramer, R., Damgård, I. B. & Nielsen, J. B. *Secure Multiparty Computation and Secret Sharing* (Cambridge University Press, New York, USA, 2015).
39. Ben-Or, M., Goldwasser, S. & Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proc. of the 20th ACM Symposium on the Theory of Computing (STOC'88)* (pp. 1–10. ACM, New York, USA, 1988).
40. Chaum, D., Crépeau, C. & Damgård, I. Multi-party unconditionally secure protocols. *Proc. of the 20th ACM Symposium on the Theory of Computing (STOC'88)* (pp. 11–19. ACM, New York, USA, 1988).
41. Maurer, U. Secure multi-party computation made simple. *Discret. Appl. Math.* **154**, 370–381 (2006).
42. Chor, B., Goldwasser, S., Micali S. & Awerbuch, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. in *Proc. of the 26th Annual Symposium on Foundations of Computer Science (FOCS'85)* 383–395 (IEEE Computer Society, Los Alamitos, California, 1985).
43. Rabin, T. & Ben-Or, M. Verifiable secret sharing and multiparty protocols with honest majority. *Proc. of the 21st ACM Symposium on the Theory of Computing (STOC'89)* (pp. 73–85. ACM, New York, USA, 1989).
44. nShield Solo HSMs, Thales Group. https://www.thalesesecurity.com/products/general-purpose-hsms/nshield-solo.
45. Hardware Security Modules, Gemalto. https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms/.
46. AWS CloudHSM, Amazon Web Services. https://aws.amazon.com/cloudhsm/.
47. Lo, H.-K., Curty, M. & Qi, B. Measurement-device-independent quantum key distribution. *Phys. Rev. Lett.* **108**, 130503 (2012).
48. Rubenok, A., Slater, J. A., Chan, P., Lucio-Martinez, I. & Tittel, W. Real-world two-photon interference and proof-of-principle quantum key distribution immune to detector attacks. *Phys. Rev. Lett.* **111**, 130501 (2013).
49. Tang, Z. et al. Experimental demonstration of polarization encoding measurement-device-independent quantum key distribution. *Phys. Rev. Lett.* **112**, 190503 (2014).
50. Tang, Y.-L. et al. Measurement-device-independent quantum key distribution over untrustful metropolitan network. *Phys. Rev. X* **6**, 011024 (2016).
51. Comandar, L. C. et al. Quantum key distribution without detector vulnerabilities using optically seeded lasers. *Nat. Photon.* **10**, 312–315 (2016).
52. Ma, C. et al. Integrated silicon photonic transmitter for polarization-encoded quantum key distribution. *Optica* **3**, 1274–1278 (2016).
53. Sibson, P. et al. Chip-based quantum key distribution. *Nat. Commun.* **8**, 13984 (2017).
54. Sibson, P. et al. Integrated silicon photonics for high-speed quantum key distribution. *Optica* **4**, 172–177 (2017).
55. Kimble, H. J. The quantum internet. *Nature* **453**, 1023–1030 (2008).
56. Castelvecchi, D. The quantum internet has arrived (and it hasn't). *Nature* **554**, 289–292 (2018).
57. Elliott, C. et al. Current status of the DARPA quantum network. In *Proc. SPIE, Quantum Information and Computation III*, Vol. 5815 (eds. Donkor, E. J., Pirich, A. R. & Brandt, H. E.) 138–149 (SPIE Press, Washington, USA, 2005).
58. Peev, M. et al. The SECOQC quantum key distribution network in Vienna. *New J. Phys.* **11**, 075001 (2009).
59. Sasaki, M. et al. Field test of quantum key distribution in the Tokyo QKD network. *Opt. Express* **19**, 10387–10409 (2011).
60. Stucki, D. et al. Long-term performance of the SwissQuantum quantum key distribution network in a field environment. *New J. Phys.* **13**, 123001 (2011).
61. Chen, T.-Y. et al. Metropolitan all-pass and inter-city quantum communication network. *Opt. Express* **18**, 27217–27225 (2010).
62. CORDIS. *China to Launch World's First Quantum Communication Network. Phys.org—News and Articles on Science and Technology*. https://phys.org/news/2017-08-china-world-quantum-network.html (2017).
63. Bennett, C. H., Brassard, G. & Robert, J. M. Privacy amplification by public discussion. *SIAM J. Comput.* **17**, 210–229 (1988).
64. Carter, J. L. & Wegman, M. N. Universal classes of hash functions. *J. Comput. Syst. Sci.* **18**, 143–154 (1979).
65. Wegman, M. N. & Carter, J. L. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.* **22**, 265–279 (1981).
66. Fitzi, M, Hirt, M. & Maurer, U. in General adversaries in unconditional multi-party computation *Proc. of the International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT'99). Lecture Notes in Computer Science*. (232–246 Springer: Berlin, 1999).
67. Renner, R. *Security of Quantum Key Distribution*. Ph.D. thesis, ETH Zurich (2005).
68. Müller-Quade, J. & Renner, R. Composability in quantum cryptography. *New J. Phys.* **11**, 085006 (2009).
69. Tehranipoor, M. & Koushanfar, F. A survey of hardware Trojan taxonomy and detection. *IEEE Des. Test. Comput.* **27**, 10–25 (2010).
70. Skorobogatov, S. & Woods, C. Breakthrough silicon scanning discovers backdoor in military chip. In *Proc. of the 14th International Conference on Cryptographic Hardware and Embedded Systems (CHES'12)* 23-40 (Springer-Verlag Berlin, Heidelberg, 2012).
71. Ben-Or, M., Crépeau, C., Gottesman, D., Hassidim, A. & Smith, A. secure multiparty quantum computation with (only) a strict honest majority. In *Proc. of the 47th Annual Symposium on Foundations of Computer Science (FOCS'06)* 249–260 (IEEE Computer Society, Los Alamitos, California, 2006).
72. Crépeau, C., Gottesman, D. & Smith, A. Secure multi-party quantum computing. *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC'02)* (pp. 643–652. ACM, New York, NY, USA, 2002).
73. Fujiwara, M. et al. Unbreakable distributed storage with quantum key distribution network and password-authenticated secret sharing. *Sci. Rep.* **6**, 28988 (2016).
74. IdQuantique, Geneve (Switzerland), http://www.idquantique.com.
75. Fitzi, M., Garay, J., Gollakota, S., Pandu Rangan, C. & Srinathan, K. Round-optimal and efficient verifiable secret sharing. *Proc. 3rd Conference on Theory of Cryptography (TCC'06)* (pp. 329–342. Springer-Verlag, Berlin, Heidelberg, 2006).
76. Gennaro, R., Ishai, Y., Kushilevitz, E. & Rabin, T. The round complexity of verifiable secret sharing and secure multicast. *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC'01)* (pp. 580–589. ACM, New York, NY, USA, 2001).
77. Shamir, A. How to share a secret. *Commun. ACM* **22**, 612–613 (1979).
78. Blakley, G. R. Safeguarding cryptographic keys. In *Proc. of the AFIPS 1979 National Computer Conference (NCC'79)* 313–317 (AFIPS Press, New Jersey, 1979). R. E. Merwin, Editor and Program Chairman.
79. Garay, J. A. & Moses, Y. Fully polynomial Byzantine agreement in $t+1$ rounds. *Proc. of the 25th ACM Symposium on the Theory of Computing (STOC'93)* (pp. 31–41. ACM, New York, USA, 1993).
80. Fischer, M. J. & Lynch, N. A. A lower bound for the tieme to assure interactive consistency. *Inf. Proc. Lett.* **14**, 183–186 (1982).
81. Hwang, W.-Y. Quantum key distribution with high loss: toward global secure communication. *Phys. Rev. Lett.* **91**, 057901 (2003).
82. Lo, H.-K., Ma, X. & Chen, K. Decoy state quantum key distribution. *Phys. Rev. Lett.* **94**, 230504 (2005).
83. Wang, X.-B. Beating the photon-number-splitting attack in practical quantum cryptography. *Phys. Rev. Lett.* **94**, 230503 (2005).