Secure Anonymous Conferencing in Quantum Networks

Federico Grasselli,[1,*] Gláucia Murta,[1,†] Jarn de Jong,[2] Frederik Hahn,[3] Dagmar Bruß,[1] Hermann Kampermann,[1] and Anna Pappa[2,4]

[1] *Institut für Theoretische Physik III, Heinrich-Heine-Universität Düsseldorf, Düsseldorf 40225, Germany*
[2] *Electrical Engineering and Computer Science Department, Technische Universität Berlin, Berlin 10587, Germany*
[3] *Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Berlin 14195, Germany*
[4] *Fraunhofer Institut for Open Communication Systems—FOKUS, Berlin 10589, Germany*

(Received 23 December 2021; accepted 26 July 2022; published 13 October 2022)

Users of quantum networks can securely communicate via so-called (quantum) conference key agreement—making their identities publicly known. In certain circumstances, however, communicating users demand anonymity. Here, we introduce a security framework for anonymous conference key agreement with different levels of anonymity, which is inspired by the ε-security of quantum key distribution. We present efficient and noise-tolerant protocols exploiting multipartite Greenberger-Horne-Zeilinger (GHZ) states and prove their security in the finite-key regime. We analyze the performance of our protocols in noisy and lossy quantum networks and compare with protocols that only use bipartite entanglement to achieve the same functionalities. Our simulations show that GHZ-based protocols can outperform protocols based on bipartite entanglement and that the advantage increases for protocols with stronger anonymity requirements. Our results strongly advocate the use of multipartite entanglement for cryptographic tasks involving several users.

DOI: 10.1103/PRXQuantum.3.040306

I. INTRODUCTION

Building on the "second quantum revolution" [1–3], quantum cryptography technologies have recently seen a rapid development both in academia and industry, with quantum key distribution (QKD) [4–6] being a prominent example [7]. The task of QKD has been generalized to multiple users with quantum conference key agreement (CKA) [8–16], where n parties establish a common secret key when linked by an insecure quantum network [17]. The key, called conference key, can subsequently be used for groupwise encryption among the n parties [18].

Besides achieving secure communication with encryption keys, an equally important privacy aspect that arises in a multipartite network is anonymity. Indeed, in numerous scenarios the communicating parties want to share a secret message and, at the same time, remain anonymous. That is, they want to keep their identities hidden from the other parties in the network, from the network manager, and even between themselves. In a world ever more demanding for privacy, many individuals would benefit from secure anonymous communication, including voters, anonymous informants, investigative reporters, secret agents, and users updating an online database (e.g., medical records).

Here, this cryptographic task is formalized as follows. Consider an n-party quantum network where, *a priori*, each party could be a participant, i.e., the sender or one of the m receivers ($m < n$) chosen by the sender. The task is to identify a set of participants and establish a secret conference key between them, such that they remain anonymous at least with respect to the other $n - m - 1$ parties. The established conference key then enables secure anonymous conferencing among the participants.

We define protocols that accomplish the described task with different anonymity requirements. In particular, an anonymous conference key agreement (ACKA) protocol reveals the participants' identities to each participant and provides them with the same conference key. Beyond that,

*federico.grasselli@hhu.de
†glaucia.murta@hhu.de

in a fully anonymous conference key agreement (fully ACKA) protocol we require that only the sender knows the number and identities of the receivers, while each receiver is only aware of their role. In both ACKA and fully ACKA protocols the participants' identities (and the conference key) are unknown to the remaining parties in the network and to a potential eavesdropper controlling the network and the quantum source.

A fully ACKA protocol could be used by a whistleblower (the sender) within a company, who wants to expose an illicit activity to some of the company's managers (the receivers). The fully ACKA protocol would ensure the anonymity of sender and receivers, thus protecting them from potential reprisals. Alternatively, an ACKA protocol could be employed by journalists to send reports from an area with limited freedom of press.

We distinguish three features characterizing the security of ACKA and fully ACKA protocols. Specifically, the protocol must be *CKA-secure*, which means that the established conference key is identical for all participants, uniformly distributed and unknown to anybody else (as in standard CKA [14,19]). The protocol must also be *anonymous*, in the sense that the identity of each participant must be kept secret from a subset of parties, depending on the required level of anonymity. Finally, the protocol must be *integrous*, that is, the identities of the sender and of the chosen receivers are correctly assigned and communicated.

In this work, we introduce rigorous security definitions for ACKA and fully ACKA, which encompass the above-mentioned security notions and are inspired by the composable security framework of QKD. Moreover, we design ACKA and fully ACKA protocols based on multipartite entangled states, namely n-party GHZ states, distributed by an untrusted source and prove their security according to our definitions.

To benchmark the performance of our GHZ-based protocols, we introduce two multiparty generalizations of the Anonymous Message Transmission protocol [20]. The protocol in Ref. [20] allows two parties to anonymously send classical messages in a network with dishonest parties, by employing bipartite private authenticated channels between every pair of parties. Such channels can be established beforehand by distributing Bell pairs and running pairwise QKD protocols. Our multiparty generalizations of the protocol in Ref. [20], named bipartite ACKA and bifully ACKA, achieve the same functionalities of ACKA and fully ACKA, respectively, while exclusively relying on bipartite entanglement (Bell pairs) shared between every pair of parties.

Our simulations, run on a quantum network like the one in Fig. 1, show that the ACKA and fully ACKA protocol based on GHZ states significantly outperform the corresponding protocol based on Bell pairs even in the finite-key regime. The advantage provided by GHZ states over Bell pairs increases with the anonymity requirements of the protocol. In the case of fully ACKA, multipartite entanglement increases the secret conference key rate by a factor proportional to the square of the number of parties in the network (n^2). Our results clearly demonstrate the benefit of multipartite entanglement for cryptographic tasks involving more than two parties and reinforce previous results supporting this claim [8,18,21].

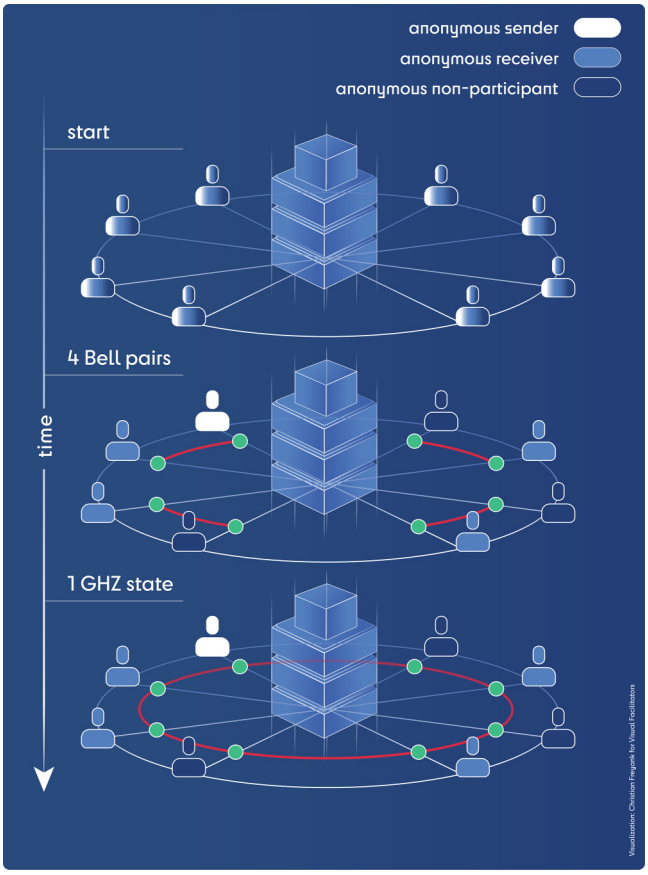

FIG. 1. Our protocols are implemented on a quantum network consisting of n parties ($n = 8$ in the picture) linked to a central quantum server. The parties' roles are not predetermined and are instead assigned during the protocol's execution. The party designated as sender selects the desired receivers to anonymously establish a shared conference key. The quantum server distributes—in each network use—either Bell pairs between distinct pairs of parties (e.g., four pairs when $n = 8$) or one GHZ state shared between all parties. While the bipartite ACKA and bifully ACKA protocols require only Bell pairs, our ACKA and fully ACKA protocols mainly use GHZ states. The figure of merit to compare our protocols is the secret conference key rate, i.e., the number of secret conference key bits generated per network use (see Sec. II C).

A. Security assumptions

The eavesdropper—called Eve—can completely control the entanglement source and the quantum network. Eve

may also listen to the public communication generated during the protocol and corrupt any subset of parties. Any party who misbehaves and willingly does not follow the prescriptions of the protocol is considered to be a corrupt party and is called *dishonest*. Eve can collaborate with the dishonest parties, having access to their private classical inputs and outputs, but is not allowed to impersonate them (e.g., store their quantum systems). Analogously to QKD, Eve has unbounded quantum power and holds a perfect quantum memory.

Conversely, the n parties are equipped with trusted quantum measurement devices and at most short-lived quantum memories. This assumption is motivated by the current state of the art in the field of quantum memories [22,23] and by the fact that most end nodes in quantum networks are simply measuring stations [24,25].

B. Relation to previous works

Early anonymous communication protocols in quantum networks [26,27] aim at transmitting classical as well as quantum information, going beyond the functionality achieved by the Anonymous Message Transmission protocol of Ref. [20]. However, such protocols [26,27] require trusted sources of multipartite entangled states, do not address secrecy, and neither tolerate dishonest network users nor noise.

These drawbacks are partially resolved by following protocols [28–30], which allow for untrusted sources of entanglement and potentially dishonest parties by introducing source-verification schemes. However, such schemes happen to be highly inefficient, technologically demanding, and are still vulnerable to noise.

Recently, the task of anonymously establishing a secret conference key, with the anonymity requirements identified as fully ACKA in our work, was addressed in Ref. [31] and further explored in Ref. [32]. The protocols in Refs. [31,32], while requiring some level of trust in the source of multipartite entangled states, also build on the same source-verification scheme of Refs. [29,33] and thus carry the same drawbacks.

In contrast to previous protocols [28–32] allowing for an untrusted albeit noiseless source, our protocols are robust against noise in the quantum network and in the untrusted source, thereby representing the first secure anonymous protocols with a *noisy* untrusted source.

Moreover, our protocols are remarkably more efficient and implementable with present-day technology, compared to previous proposals [26–32]. Indeed, they verify the entangled states prepared by the source over a small fraction of states through local measurements, leaving the majority of the states for establishing the conference key. Previous protocols, instead, either assume a noiseless trusted source (which is a strong assumption) [26,27], or consume the majority of the entangled states for verification (resulting in asymptotically null key rates even in a noiseless scenario) [29–32], or require multiqubit gates and perfect quantum memories [28], which are not yet technologically available.

Regarding security, unlike any previous anonymous quantum protocol, we prove the security and anonymity of our protocols in the experimentally relevant regime of finite keys, by extending the ε-security framework of QKD to include anonymity and integrity.

Altogether, the efficiency, noise tolerance, and ε-security of our anonymous protocols allow us to compute nonzero conference key rates in noisy and lossy quantum networks, both in the asymptotic and finite-key regimes. We emphasize that such a result could not be obtained in any previous anonymous-communication protocol based on multipartite entanglement for the reasons outlined above.

C. Use of the anonymous conference key

Contrary to standard QKD and CKA, the secret conference key established by ACKA and fully ACKA protocols must be used in a nontrivial encryption scheme in order to protect the participants' identities. If the sender simply encrypts a message with the conference key and broadcasts it, every other party would identify the sender as the party who performed the broadcast.

In the ACKA scenario, the solution is straightforward. Every party in the network is asked to broadcast a random string, while the sender broadcasts the encrypted message. Only the receivers are able to decrypt the sender's broadcast by using the conference key.

In the fully ACKA scenario, the receivers do not know the sender's identity and the previous solution could reveal it. In this case, a possibility would be to employ classical anonymous broadcasting protocols [34].

II. RESULTS

A. Security definitions

Our approach to define the security of ACKA and fully ACKA protocols starts from the identification of three properties that an ideal protocol is expected to satisfy.

The first property that we require is integrity. At the beginning of an ACKA (fully ACKA) protocol, the participants are not yet determined. Therefore, the first step of such protocols consists in running an identity-designation (ID) subprotocol. The ID subprotocol determines the sender, notifies the receivers of their roles and, in the case of ACKA, notifies the receivers of the other participants' roles. A protocol is integrous if its ID subprotocol works perfectly, i.e., if either it correctly communicates the roles

of sender and receivers, or it aborts for every party in the network.

Conditioned on the fact that the ID subprotocol does not abort and correctly assigns the identities, we require the ideal protocol to be CKA-secure. That is, either it outputs the same random conference key for every participant, uncorrelated from any information held by the nonparticipants and Eve, or it aborts from the point of view of the participants.

Finally, an ideal protocol should be anonymous, i.e., the identity of each participant must be kept secret from the nonparticipants and Eve—as well as from the other receivers in the case of fully ACKA.

We remark that dishonest participants may broadcast their identity—or the identity of all the participants in the case of ACKA. Hence, in the presence of dishonest participants we cannot impose the same anonymity requirements. Similarly, no CKA-security is required if some participants are dishonest, as they may publicly reveal the secret conference key.

Inspired by the composable security framework of QKD [35,36], we define the security of ACKA and fully ACKA protocols from the output state of the protocol—i.e., the state of the classical and quantum registers of each party, including the eavesdropper, at the end of the protocol. More specifically, we introduce a security definition, which quantifies, for every property, how close the output state is to a state with the required property. Informally, our security definition can be stated as follows (the rigorous definition, Definition 6, is given in Appendix B).

Definition 1 (Security (informal)): An ACKA (fully ACKA) protocol is ε-secure, with $\varepsilon = \varepsilon_{\text{in}} + \varepsilon_{\text{CKA}} + \varepsilon_{\text{an}}$, if it satisfies the following three conditions:

(a) ε_{in}-integrity: the ID subprotocol correctly assigns the roles of sender and receivers or aborts for every party in the network, except for a probability smaller than ε_{in}.
(b) ε_{CKA}-CKA-security: in the case of honest participants, conditioned on the ID subprotocol correctly assigning the participants' identities, the output state is ε_{CKA}-close to the output state of a protocol that either delivers the same secret conference key to every participant, or aborts for every participant.
(c) ε_{an}-anonymity for ACKA: the output state of any subset of nonparticipants and Eve is ε_{an}-close to a state, which is independent of the identity of the remaining parties. ε_{an}-anonymity for fully ACKA: the output state of any subset of parties (except for the sender) and Eve is ε_{an}-close to a state, which is independent of the identities of the other parties.

In Sec. II B we introduce our ACKA and fully ACKA protocols based on GHZ states shared by all parties. Motivated by the fact that our fully ACKA protocol does not satisfy the (strong) anonymity condition of Definition 1 but still retains important anonymity features, we provide a weaker anonymity definition satisfied by our fully ACKA protocol.

The reason for which the fully ACKA protocol cannot satisfy the strong anonymity condition is that the receivers, by executing the protocol, gain access to information (e.g., whether the protocol aborts or not) that can depend on the identities of the other participants. Such a dependence can occur if the untrusted source distributes asymmetric states, which are noisier for some parties than for others, instead of the permutationally symmetric GHZ states. Similar problems affecting anonymity were already mentioned in Ref. [28] regarding the protocol proposed in Ref. [37].

However, we emphasize that honest-but-curious receivers may be able to deduce the identity of other participants only if they combine the identity-dependent information obtained from the protocol with a detailed knowledge of the asymmetric states distributed by the source—or with any other asymmetric specification of the protocol causing identity-dependent events. Therefore, the anonymity of the parties can be preserved if honest-but-curious receivers do not have access to the asymmetric specifications of the protocol's implementation (e.g., if they are secret or if the publicly available specifications are symmetric), even if the source actually distributes asymmetric states. Note, however, that we cannot prevent dishonest parties or Eve from broadcasting the actual, asymmetric, specifications of the protocol's implementation at any point in time, thus jeopardizing anonymity with respect to honest receivers.

This weaker version of anonymity is captured by the following definition, where *w* abbreviates "weak" and *(n)p* abbreviates "(non)participant." The definition is satisfied by our fully ACKA protocol with GHZ states (the formal definition is given in Appendix B).

Definition 2 (Weak anonymity for fully ACKA (informal)): A fully ACKA protocol is ε_{Wan}-weak-anonymous, with $\varepsilon_{\text{Wan}} = \varepsilon_{\text{NPan}} + \varepsilon_{\text{Pan}}$, if the following two conditions are satisfied:

1. (Anonymity with respect to nonparticipants and Eve.) The output state of the protocol satisfies the ε-anonymity condition for ACKA protocols with $\varepsilon = \varepsilon_{\text{NPan}}$.
2. (Anonymity with respect to honest-but-curious receivers.) When the protocol's specifications known to the parties are symmetric—i.e., invariant under permutations of parties—any subset of honest-but-curious receivers cannot guess the identity of other participants with a higher probability

than the trivial guess, except for a small deviation $\varepsilon_{\mathrm{Pan}}$.

B. ACKA and fully ACKA protocols

Here we present an ACKA and a fully ACKA protocol, which rely on the multipartite correlations of GHZ states shared by all the parties in the network. However, we emphasize that the untrusted source could prepare completely arbitrary states and potentially be controlled by Eve. Similarly, any dishonest party can behave differently from the actions prescribed by the protocol. Yet, our ACKA and fully ACKA protocols are secure in the sense of Definition 1.

For convenience, we identify the sender as Alice and the intended receivers as Bob_l, for $l \in \{1, \ldots, m\}$, where the number of receivers m is not predetermined and can be chosen by Alice during the protocol. Both our protocols require the following:

1. A shielded laboratory for each party, equipped with a trusted measurement device, a trusted postprocessing unit and a private source of randomness.
2. A bipartite private channel for each pair of the n parties. We obtain it with a bipartite public authenticated channel and a shared secret key to encrypt the communication over the channel (with one-time pad). Hence, the communication over the bipartite private channel is secret (i.e., only known to the legitimate parties), whereas the identities of the parties using the channel are public.
3. An authenticated broadcast channel.
4. A public source of randomness that is not controlled by the adversary.

The ACKA protocol also requires previously shared conference keys among every subset of parties in the network (note that this requirement could be dropped by introducing a minor overhead in the number of bipartite private channel uses). Part of these keys are consumed during the execution of the ACKA protocol, which is thus a key-growing algorithm.

In the following, we provide a high-level description of the steps of our ACKA and fully ACKA protocols, whose core consists in measuring GHZ states to anonymously generate a shared conference key. The additional steps required to ensure integrity and CKA-security (namely the ID subprotocol, error correction, and privacy amplification) are summarized here and further detailed in Appendices C and D, together with more exhaustive protocol descriptions.

In order to ensure that the classical communication required by ACKA and fully ACKA is anonymous, we make use of classical subroutines introduced in Ref. [20]—specifically Parity, Veto, and Collision Detection—which are run on the bipartite private channels. Note that, in our simulations, we account for the generation of the secret keys—needed to implement the bipartite private channels—by running pairwise QKD protocols over distributed Bell pairs. As the name suggests, the Parity protocol computes the Parity of the input bits while preserving the anonymity of the parties.

Protocol 1: Anonymous conference key agreement (**ACKA**).

---

1. The parties run a subprotocol, called identity designation for ACKA (ACKAID, Protocol 6 in Appendix C), after which Alice is established to be the sender and Bob_l, for $l \in \{1, \ldots, m\}$, the receivers.
2. Alice and the Bobs recover a pre-established conference key.
3. Alice generates a random bitstring, called testing key, where 1 corresponds to a test round and 0 to a key generation round (each bit equals 1 with probability p). Alice broadcasts a compressed version of the testing key, encrypted (with one-time pad) with a portion of the pre-established conference key of step 2, so that each Bob can decrypt it and recover the testing key. All the other parties broadcast a random string of the same length.
4. Repeat for L rounds:
   4.1. An untrusted source distributes a state to each of the n parties. Ideally, the source prepares an n-party GHZ state.
   4.2. Alice and the Bobs measure their qubits according to the testing key. They measure in the (Pauli) Z basis if the round is a key generation round, or in the (Pauli) X basis if the round is a test round. All the other parties measure X. The outcomes $+1$ and -1 of each Pauli measurement are mapped to the binary values 0 and 1, respectively. The Z outcomes of each participant form their raw conference key.
5. Once all the qubits have been measured (in case the parties hold short-lived quantum memories, we wait for a time longer than their coherence time), the testing key is anonymously revealed to all parties by iterating the Parity protocol (Protocol 3).
6. For every round labeled as a test round, the n parties perform the Parity protocol with the following inputs: every party, except for Alice, inputs the outcome of their X measurement while Alice inputs a random bit. By combining the outputs of Parity with her test-round outputs, Alice computes Q_X^{obs}, which is the fraction of test rounds where the X outcomes of the n parties have parity 1.
7. Verification of secrecy: Alice compares Q_X^{obs} with the predefined value Q_X. If $Q_X^{\mathrm{obs}} + \gamma(Q_X^{\mathrm{obs}}) > Q_X +$

$\gamma(Q_X)$, where $\gamma(Q_X)$ is the statistical fluctuation, Alice concludes that the verification failed.

8. Error correction (ACKAEC, Protocol 9): Alice anonymously broadcasts error-correction information based on a predefined value Q_Z for the pairwise error rate between the Z outcomes of Alice and of each Bob. The Bobs use the information to correct their raw keys and verify that they match Alice's raw key. Alice's broadcast is encrypted with the pre-established conference key and only the Bobs can decrypt it. If the error correction or the verification of secrecy (step 7) failed, the participants abort the protocol, but this information is encrypted and only available to them.
9. Privacy amplification (PA): the public randomness outputs a two-universal hash function. Alice and each Bob apply the two-universal hash function on their error-corrected keys and obtain the secret conference keys of length ℓ.

In the fully ACKA scenario, the participants do not know each other's identity, hence they cannot use pre-established conference keys to share the testing key as in the ACKA protocol (Protocol 1). Therefore, in the fully ACKA protocol we introduce a subprotocol called the Testing Key Distribution (TKD) protocol (Protocol 8 in Appendix C), which allows Alice to anonymously distribute the testing key to the Bobs.

Protocol 2: Fully Anonymous conference key agreement (**fully ACKA**)

1. The parties perform the fully ACKAID subprotocol (Protocol 7), after which Alice is established to be the sender and Bob_l, for $l \in \{1,\dots,m\}$, the receivers.
2. Alice generates a random bitstring, called testing key, where 1 corresponds to a test round and 0 to a key generation round (each bit equals 1 with probability p). Additionally, Alice generates the bitstrings $\vec{r}_l$ (for $l \in \{1,\dots,m\}$), which are later used to encrypt some communication between Alice and each Bob.
3. The parties perform the TKD protocol (Protocol 8) in order for Alice to distribute the testing key and the string $\vec{r}_l$ to the corresponding Bob_l.
4. Repeat for L rounds:
   4.1. An untrusted source distributes a state to each of the n parties. Ideally, the source prepares an n-party GHZ state.
   4.2. Alice and the Bobs measure their qubits according to the testing key. They measure in the Z basis if the round is a key generation round, or in the X basis if the round is a test round. All the other parties measure X. The Z outcomes of each participant form their raw conference key.
5. Once all the qubits have been measured, the testing key is anonymously revealed by iterating the Parity protocol (Protocol 3).
6. For every round labeled as a test round, the n parties perform the Parity protocol with the following inputs: every party, except for Alice, inputs the outcome of their X measurement while Alice inputs a random bit. By combining the outputs of Parity with her test-round outputs, Alice computes Q_X^{obs}, which is the fraction of test rounds where the X outcomes of the n parties have parity 1.
7. Verification of secrecy: Alice compares Q_X^{obs} with the predefined value Q_X. If $Q_X^{\text{obs}} + \gamma(Q_X^{\text{obs}}) > Q_X + \gamma(Q_X)$, where $\gamma(Q_X)$ is the statistical fluctuation, Alice concludes that the verification failed. If the verification of secrecy failed, or if a party detected any malfunctioning in the TKD protocol (step 3), the protocol aborts for every party.
8. Error correction (fully ACKAEC, Protocol 10): Alice anonymously broadcasts error-correction information based on a predefined value Q_Z for the pairwise error rate between the Z outcomes of Alice and of each Bob. The Bobs use the information to correct their raw keys and verify that they match Alice's raw key. If the error correction fails, the protocol aborts but this information is encrypted with the strings $\vec{r}_l$ and thus only available to Alice and the Bobs.
9. Privacy amplification (PA): the public randomness outputs a two-universal hash function. Alice and each Bob apply the two-universal hash function on their error-corrected keys and obtain the secret conference keys of length ℓ.

In the Supplemental Material [38] we prove the security of Protocols 1 and 2 according to the formal statement of Definition 1, provided in Appendix B (Definition 6). The security claims of the protocols are reported in the following theorem.

Theorem 3 (Security): The ACKA protocol based on GHZ states (Protocol 1) yields a secret conference key of net length

$$\ell_{\text{net}} = L(1-p)\left[1 - h\left(Q_X + \gamma(Q_X)\right) - h(Q_Z)\right] - \log_2 \frac{2(n-1)}{\varepsilon_{\text{EC}}} - 2\log_2 \frac{1}{2\varepsilon_{\text{PA}}} - Lh(p) - n, \quad (1)$$

and is ε_{tot}-secure according to Definition 1, with $\varepsilon_{\text{tot}} = 2^{-r_V} + (n-1)\varepsilon_{\text{enc}} + 2\varepsilon_x + \varepsilon_{\text{EC}} + \varepsilon_{\text{PA}}$ and where $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function.

The fully ACKA protocol based on GHZ states (Protocol 2) yields a secret conference key of length

$$\ell = L(1-p)\left[1 - h\left(Q_X + \gamma(Q_X)\right) - h(Q_Z)\right] - \log_2 \frac{2(n-1)}{\varepsilon_{\mathrm{EC}}} - 2\log_2 \frac{1}{2\varepsilon_{\mathrm{PA}}}, \tag{2}$$

and is $\varepsilon_{\mathrm{tot}}$-secure according to Definition 1 but with the anonymity condition replaced by Definition 2, with $\varepsilon_{\mathrm{tot}} = 2^{-(r_V-2)} + (n-1)(6\varepsilon_{\mathrm{enc}} + 2^{-(r_N-1)}) + \varepsilon_{\mathrm{EC}} + 6\varepsilon_x + 3\varepsilon_{\mathrm{PA}}$.

Since Protocol 1, differently from Protocol 2, is a key-growing algorithm, in Eq. (1) we reported the net key length after one run of the protocol, which is obtained by subtracting from ℓ the number of consumed bits of pre-established conference keys. The protocols' parameters appearing in the conference key length and in the security parameter $\varepsilon_{\mathrm{tot}}$ are specified in Table I of Appendix D.

C. Performance comparison

In order to assess the performance of Protocols 1 and 2 and the benefit of GHZ states in anonymously establishing a conference key, we design protocols achieving the same tasks without resorting to multipartite entanglement. The protocols, which use only bipartite private channels hence named bipartite ACKA and bifully ACKA, are generalizations of the Anonymous Message Transmission protocol [20] to more than two parties. In Appendix E we provide a detailed description of bipartite ACKA (Protocol 11) and bifully ACKA (Protocol 12) along with their security claims according to Definition 1.

All the protocols are run on the same quantum network (see Fig. 1). We model the network of n parties as a star-shaped network where every party is linked to an untrusted quantum server—potentially operated by Eve—by an equally lossy quantum channel of transmittance η.

The quantum server is programmed to distribute either the n-party GHZ state $|\mathrm{GHZ}_n\rangle = (|0\rangle^{\otimes n} + |1\rangle^{\otimes n})/\sqrt{2}$, which is used to extract the conference key in ACKA and fully ACKA, or the Bell state $|\mathrm{GHZ}_2\rangle$ to every pair of parties in order to implement the bipartite private channels by running BB84 protocols [4].

We assume that both states are encoded in some binary degree of freedom of single photons (e.g., polarization), such that the probability of detecting a (potentially noisy) Bell pair and GHZ state is η^2 and η^n, respectively. We allow for noisy states due to a faulty state preparation by the quantum server, which prepares the states by applying CNOT gates with failure probability f_G. This leads to nonzero error rates Q_X and Q_Z of the GHZ state and Q_{Xb} and Q_{Zb} of the Bell pairs. We refer the reader to the Supplemental Material [38] for the exact relation between the error rates and f_G.

We compare the four protocols—ACKA versus bipartite ACKA, and fully ACKA versus bifully ACKA—in terms of their conference key rate, that is, the number of secret conference bits shared by Alice and the intended Bobs per network use. We define one network use to be the preparation and attempted distribution of photons from the quantum server to the parties, regardless of whether the photons are lost or not. We assume that each channel between the server and a party can transmit at most one photon per network use. However, multiple photons can be transmitted in parallel from the server to the parties in a single network use, namely entangled in a GHZ state shared between all parties or in multiple Bell pairs shared between disjoint pairs of parties (see Fig. 1).

The conference key rate of the ACKA (fully ACKA) protocol reads $r = \ell_{\mathrm{net}}/L_{\mathrm{tot}}$ ($r = \ell/L_{\mathrm{tot}}$), where ℓ_{net} (ℓ) is given in Eq. (1) [respectively, Eq. (2)] and L_{tot} is the total number of network uses. Recall that the ACKA and fully ACKA protocols employ GHZ states (for key generation) and Bell pairs (for the bipartite private channels), both contributing to the total number of network uses L_{tot}. However, the contribution of Bell pairs is marginal since, in the majority of the rounds, the parties are generating conference key bits by measuring a GHZ state. Conversely, all the network uses in bipartite ACKA and bifully ACKA are devoted to the distribution of Bell pairs, which are also used to establish the conference key.

In Fig. 2 we plot the conference key rates of the four protocols in the finite-key regime. The key rates are numerically optimized over the protocols' parameters for each value of L_{tot}, having fixed the security parameter (Definition 1) to $\varepsilon_{\mathrm{tot}} = 10^{-8}$.

In the optimizations, we set the CNOT gate failure probability to $f_G = 0.02$ and to $f_G = 0.01$ [41], while the transmittance is given by $\eta = 10^{-\gamma d/10}$, where d is the length of the channel party server and $\gamma = 0.17$ dB/km (each channel is assumed to be an ultralow-loss fiber). We set the distance to $d = 2$ km and $d = 10$ km to simulate common metropolitan communication scenarios. Further details on the conference key rates and their optimization are given in the Supplemental Material [38].

From Fig. 2 we deduce that the ACKA and fully ACKA protocols, based on GHZ states, can yield higher conference key rates than the protocols exclusively based on Bell pairs (bipartite ACKA and bifully ACKA), especially in the high-L_{tot} regime, where finite-key effects can be neglected. In order to understand the intrinsic reason behind this, we compute the asymptotic conference key rates of our anonymous protocols. Indeed, the asymptotic key rates are devoid of statistical corrections and are independent of the subprotocols requiring a fixed number of network uses regardless of the key length (e.g., Veto or ID). Hence, they reveal the bare scaling of the protocols' performance with respect to the number of parties and the quality of the source states.

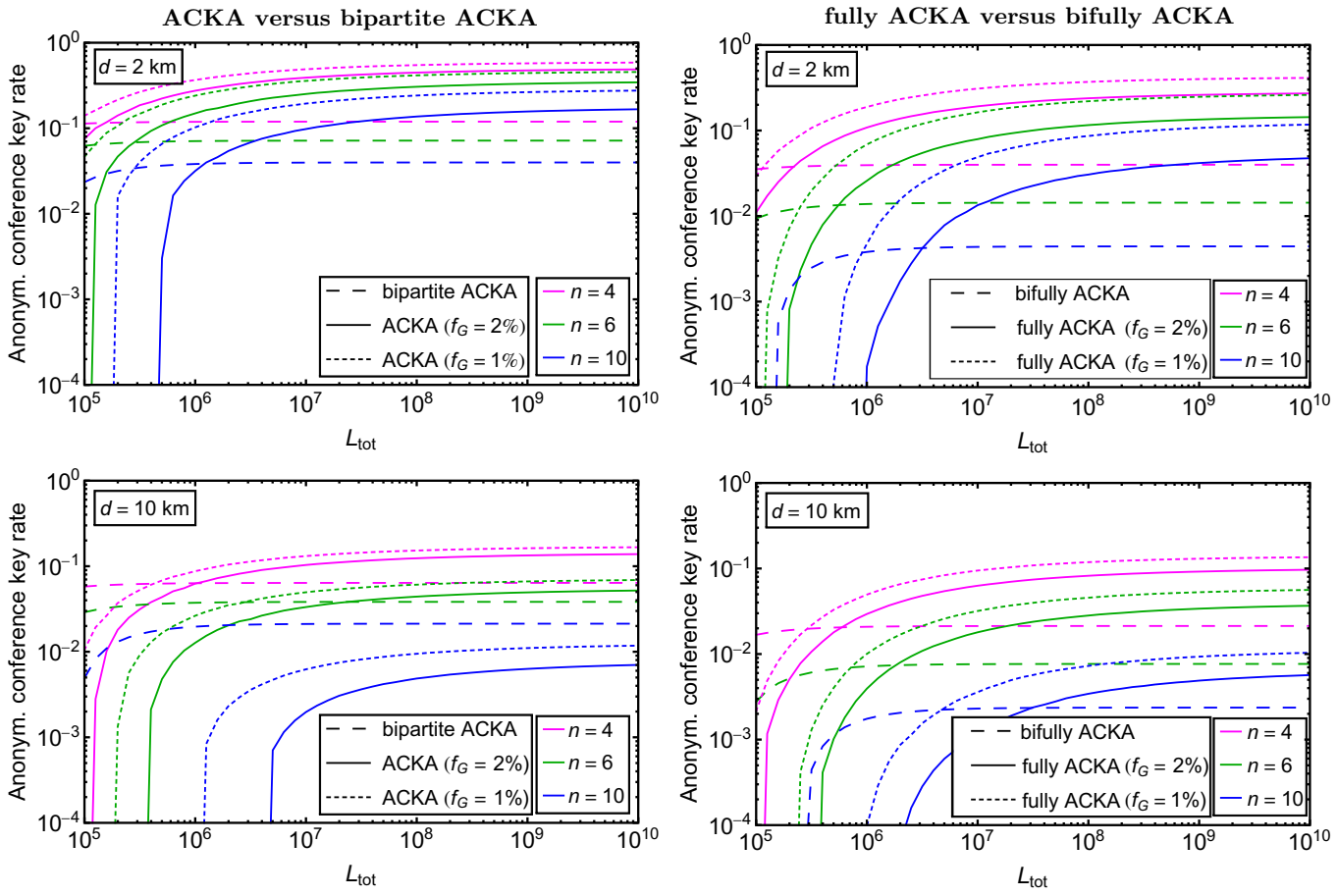

FIG. 2. Secret conference key rates of our anonymous protocols as a function of the total number of network uses (L_{tot}), for different numbers of parties in the network (n). The security parameter is fixed to $\varepsilon_{\text{tot}} = 10^{-8}$. The protocols based on GHZ states (ACKA and fully ACKA) outperform the ones based on Bell pairs (bipartite ACKA and bifully ACKA) already for a relatively low number of network uses. In our network model a quantum server prepares either GHZ states or Bell pairs by repeatedly applying a faulty CNOT gate with failure probability f_G. The qubits are encoded on single photons and distributed to the equally distanced parties via ultralow-loss fiber. The gate failure probability is either fixed to $f_G = 0.02$ or $f_G = 0.01$ (the difference in the rates of bipartite ACKA and bifully ACKA is negligible, hence we report only the $f_G = 0.02$ case) and the distance party server is fixed to $d = 2$ km (top) and $d = 10$ (bottom).

The asymptotic conference key rates of the ACKA and bipartite ACKA protocol are given by

$$r^{\infty} = \lim_{L_{\text{tot}}\to\infty} r = \eta^n \left[1 - h(Q_X) - h(Q_Z)\right], \tag{3}$$

$$r_b^{\infty} = \lim_{L_{b\text{tot}}\to\infty} r_b = \frac{\lfloor n/2 \rfloor\, \eta^2 \left[1 - h(Q_{Xb}) - h(Q_{Zb})\right]}{n(n-1)}, \tag{4}$$

respectively, while the asymptotic conference key rates of the fully ACKA and bifully ACKA protocol read

$$r_f^{\infty} = \lim_{L_{f\,\text{tot}}\to\infty} r_f = \frac{\eta^n \left[1 - h(Q_X) - h(Q_Z)\right]}{1 + n(n-1)\eta^{n-2} h(Q_Z)/\{\lfloor n/2 \rfloor \left[1 - h(Q_{Xb}) - h(Q_{Zb})\right]\}}, \tag{5}$$

$$r_{bf}^{\infty} = \lim_{L_{bf\,\text{tot}}\to\infty} r_{bf} = \frac{\lfloor n/2 \rfloor\, \eta^2 \left[1 - h(Q_{Xb}) - h(Q_{Zb})\right]}{n(n-1)^2}. \tag{6}$$

For completeness, we also report the asymptotic conference key rate of a standard CKA protocol (namely, the multipartite BB84 protocol introduced in Ref. [9]), which distils a secret conference key for the n parties in the network by distributing single photons entangled in GHZ states:

$$r^{\infty}_{\mathrm{CKA}} = \eta^n \left[1 - h(Q_X) - h(Q_Z)\right]. \tag{7}$$

Interestingly, we note that the above rate is identical to the asymptotic rate of the ACKA protocol (3), even though the latter protocol is more involved as it guarantees anonymity for the participants.

Even in the case of standard CKA, we can devise an alternative protocol, which uses only Bell pairs to establish the conference key. Note that such a protocol would require at least two network uses for every shared conference key bit, regardless of the number of parties. For instance, in the case of four parties—Alice and three Bobs—the first network use distributes Bell states to Alice-Bob_1 and Bob_2-Bob_3, while the second network use distributes a Bell pair to Bob_1-Bob_2. By employing a one-time pad the n parties can establish a shared conference key from their pairwise secret keys. If we then consider that the Bell pairs distributed by the source can be noisy or lost, the asymptotic key rate of a CKA protocol exclusively based on Bell pairs is independent of n and reads

$$r^{\infty}_{b\mathrm{CKA}} = \frac{1}{2}\eta^2 \left[1 - h(Q_{Xb}) - h(Q_{Zb})\right]. \tag{8}$$

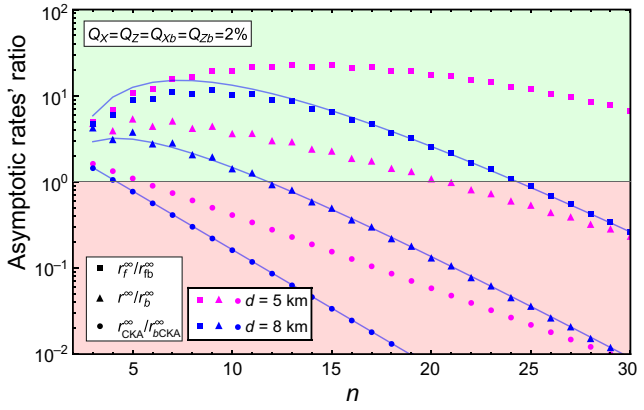

FIG. 3. Ratios between the asymptotic ($L_{\mathrm{tot}} \to \infty$) secret conference key rates of the fully ACKA (squares), ACKA (triangles), and CKA (circles) protocols with GHZ states and the rates of the corresponding protocols with Bell pairs, as a function of the number of parties in the network. The ratios of ACKA and fully ACKA are well into the green region (i.e., greater than one), indicating that the use of GHZ states is advantageous compared to using only Bell pairs, when anonymity requirements are added to a standard CKA scheme. The error rates of the GHZ (Q_Z, Q_X) and Bell states (Q_{Zb}, Q_{Xb}) are fixed to 2%, the distance party server to $d = 5$ km and $d = 8$ km. The blue lines are given by Eq. (9) with $d = 8$ km.

In Fig. 3 we plot the ratios r^{∞}/r_b^{∞}, $r_f^{\infty}/r_{bf}^{\infty}$ and $r^{\infty}_{\mathrm{CKA}}/r^{\infty}_{b\mathrm{CKA}}$ as a function of the number of parties n, for a distance party server of $d = 5$ km and $d = 8$ km. We fix the error rates of the GHZ state and Bell state to the same value (2%), rather than modeling the state preparation with CNOT gates, to highlight implementation-independent behaviors. The ratios for the task of ACKA and fully ACKA are well above 1 and can comfortably exceed one order of magnitude, indicating the advantage of using GHZ states over Bell pairs to anonymously establish conference keys. Conversely, the use of GHZ states barely brings any benefit for standard CKA, at least in our network model where the server can simultaneously distribute several Bell pairs (this is not necessarily true when considering different network models [8]).

III. DISCUSSION

We introduced a security definition that reflects all the desired properties of an anonymous conference-key-agreement protocol and encompasses different levels of anonymity. Even though our security definition is inspired by the composable security paradigm, it remains an open point whether the three conditions given in Definition 1 imply composability for an ACKA (fully ACKA) protocol.

We designed efficient and noise-robust protocols exploiting the multipartite entanglement of GHZ states, proved their security according to our security definition, and benchmarked their performance with counterpart protocols exclusively relying on the bipartite entanglement of Bell pairs.

Our security proofs rely on the assumption that the n parties either hold a short-lived quantum memory or have no quantum memory at all (bounded storage model [42]). The eavesdropper, instead, holds a perfect quantum memory and can perform coherent attacks as in standard QKD. While the bounded storage assumption is crucial for the security of ACKA and fully ACKA, it is irrelevant for the bipartite ACKA and bifully ACKA protocols, which would still remain secure even if this assumption is dropped. Potentially, there might exist protocols based on multipartite entanglement that are secure without the bounded storage assumption for the n parties, and at the same time retain the valuable properties of efficiency, noise robustness (untrusted source), and anonymity featured by our ACKA and fully ACKA protocols. However, we think that there could be a fundamental reason forbidding the existence of such protocols. A hint in this direction comes from the impossibility of quantum bit commitment [43]. Indeed, our ACKA and fully ACKA protocols essentially require each nonparticipant to commit to a bit for each distributed state, where the bit is their X measurement outcome, and to reveal the bit when the state distribution is over. Thus, quantum bit commitment can be viewed as a resource in our protocols, which, however, can only

be implemented when other assumptions (e.g., bounded storage) are made.

From the plots of Fig. 2, we observe that the protocols based on GHZ states (ACKA and fully ACKA) outperform the protocols exclusively based on Bell pairs (bipartite ACKA and bifully ACKA) for an experimentally feasible [18] number of network uses, starting from values as low as $L_{\text{tot}} = 10^5$. However, increasing the distance between the parties and the source of entanglement is more detrimental for the protocols based on GHZ states (bottom plots), due to the higher probability of losing at least a photon in a GHZ state compared to a Bell state. This effect can be partially mitigated if the preparation quality of the entangled states is improved (dotted lines in Fig. 2). Indeed, reducing the gate failure probability from 2% to 1% significantly impacts the rates of the protocols based on GHZ states, while leaving the rates of the Bell-state protocols almost unchanged—hence in Fig. 2 we reported only the $f_G = 2\%$ case for the protocols based on Bell pairs.

Moreover, we observe that the protocols based on GHZ states require a higher number of network uses to yield a nonzero key rate, compared to the protocols based on Bell states. This is partially explained by the fact that we did not include finite-key effects in the rate at which the bipartite private channels distribute secret bits, while we performed a full finite-key analysis for the conference key rate with GHZ states. Since the bipartite ACKA and bifully ACKA protocols rely on the bipartite private channels much more prominently than ACKA and fully ACKA, the former protocols are advantaged compared to the latter for low numbers of network uses. Even if we performed a full finite-key analysis for the bipartite ACKA and bifully ACKA rates, they would still outperform the rates of ACKA and fully ACKA in the low-L_{tot} regime (see Supplemental Material [38] for a detailed discussion).

Finite-key effects aside, Fig. 3 clearly displays the benefit of employing multipartite entanglement, in the form of GHZ states, over Bell pairs, for the cryptographic tasks of ACKA and fully ACKA. Interestingly, the superiority of GHZ-based protocols does not increase monotonically with n and instead displays an optimal value of n for which GHZ states are most beneficial. This is due to the interplay between two effects.

On the one hand, establishing one conference key bit while maintaining *anonymity* requires only one GHZ state in both ACKA and fully ACKA, while it requires $n(n-1)$ Bell pairs in the case of bipartite ACKA and $n(n-1)^2$ Bell pairs in bifully ACKA (due to the iteration of Parity protocols). This is only partially mitigated by the parallel distribution of Bell pairs from the source in each network use, which compensates by a factor $\lfloor n/2 \rfloor^{-1}$. This effect dominates at low n and causes the ratios of ACKA and fully ACKA to increase with n. On the other hand, when the number of parties—and hence the number of simultaneously transmitted photons in a GHZ state—increases, the rate of the GHZ-based protocols exponentially decreases due to photon loss with a factor η^n, while the rates of bipartite ACKA and bifully ACKA present a constant factor η^2. Overall, the ratios exponentially decrease at high n.

The interplay between a polynomial increase due to the efficiency of GHZ states and an exponential suppression due to photon loss becomes clear when computing the ratios, having set the error rates to zero and ignoring the floor functions:

$$\begin{aligned} \frac{r^{\infty}_{\text{CKA}}}{r^{\infty}_{b\text{CKA}}} &\sim 2\eta^{n-2}, \\ \frac{r^{\infty}}{r^{\infty}_{b}} &\sim 2(n-1)\eta^{n-2}, \\ \frac{r^{\infty}_{f}}{r^{\infty}_{bf}} &\sim 2(n-1)^2\eta^{n-2}. \end{aligned} \tag{9}$$

The above functions are plotted as solid lines in Fig. 3 for $d = 8$ km and reproduce the scaling of the plot points well.

From Eq. (9) we conclude that, starting from a standard CKA scenario, if we require the participants to be anonymous with respect to the other parties and to Eve (ACKA scenario), the conference key rate of a GHZ-based protocol gains a factor of $n-1$ over the rate of a protocol based on Bell pairs. If we additionally require anonymity among the participants (fully ACKA scenario), the key rate of a GHZ-based protocol gains a factor of $(n-1)^2$ compared to just using Bell pairs. This holds despite allowing the source to simultaneously distribute multiple Bell pairs in one network use. Overall, this suggests that adding anonymity requirements significantly increases the advantage of multipartite entanglement over bipartite entanglement.

Furthermore, we point out that the exponential suppression in Eq. (9) due to photon loss is heavily dependent on the length in kilometers d of each party-server channel. This implies that the maximum number of parties n for which the GHZ-based protocols provide an advantage (the ratios are above one) increases rapidly as d decreases, as suggested by Fig. 3. Indeed, for applications where the distance between parties is of few kilometers, the use of GHZ states is still advantageous in networks of more than 100 parties (see discussion in Supplemental Material [38]). Additionally, the suppression due to photon loss could be avoided altogether by resorting to a different class of multipartite entanglement, namely W states [44]. Indeed, it has been shown that CKA can also be achieved when the parties share a W state postselected after single-photon interference [10]. This means that only one out of n photons needs to be successfully transmitted, yielding a

conference key rate that scales with η (instead of η^n when using GHZ states).

The performance advantage provided by GHZ states in the fully ACKA scenario, however, comes at the expense of a slightly weaker anonymity claim with respect to honest-but-curious receivers. Indeed, the fully ACKA protocol is anonymous according to Definition 2 while the bifully ACKA protocol satisfies Definition 1. This is the result of a trade-off with the robustness to noise and efficiency featured by the fully ACKA protocol. Indeed, the fact that our protocol efficiently verifies the source state only in a small fraction of rounds and is robust against noisy state preparations, allows it to succeed even when asymmetries affect the multipartite states distributed to the parties. On one hand, this makes our protocol very efficient, noise robust, and practical, as opposed to previous anonymous protocols based on multipartite entanglement [26–32]. On the other hand, it prevents the protocol from satisfying the strong anonymity condition when asymmetric multipartite states are distributed by the source, as already discussed in Sec. II A.

In summary, our work identifies the anonymous communication tasks of ACKA and fully ACKA, provides them with a rigorous security framework, and demonstrates that the multipartite quantum correlations of GHZ states can increase the rate at which conference keys are anonymously generated within a quantum network, compared to solely relying on bipartite entanglement. The gain in the rate increases as anonymity requirements are added to the protocol, it can comfortably exceed one order of magnitude, and scales with the square of the number of parties in the network ($\sim n^2$) in the case of a fully ACKA protocol. This is a striking result when compared to previously known scaling improvements due to multipartite entanglement [8,18,21], which expected at most a linear gain in the rate scaling ($\sim n$). Moreover, previous results apply in the case of particularly favorable network structures (e.g., networks with bottlenecks), whereas in this work we consider a network that is symmetric and does not privilege any party. Additionally, we obtain such a scaling advantage despite the fact that the simultaneous distribution of multiple Bell pairs and of a single GHZ state contribute equally to the count of network uses, differently from previous works.

Therefore, our results provide strong evidence for the superiority of multipartite entanglement over bipartite entanglement for multiuser cryptographic tasks and pave the way for the implementation of quantum communication protocols beyond QKD and CKA.

ACKNOWLEDGMENTS

We thank Lennart Bittel and Nathan Walk for fruitful discussions. F.G., G.M., D.B., and H.K. acknowledge support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1-390534769. D.B. and H.K. acknowledge support by the QuantERA project QuICHE, via the German Ministry for Education and Research (BMBF Grant No. 16KIS1119K). A.P. and J.d.J. acknowledge support from the German Research Foundation (DFG, Emmy Noether Grant No. 418294583) and F.H. from the German Academic Scholarship Foundation.

F.G. and G.M. contributed equally to this work.

APPENDIX A: NOTATION

Here we describe the formalism and notation used in the Appendices and the Supplemental Material [38].

(a) Strings of numbers and bitstrings are denoted with the vector sign $\vec{\ }$, while symbols that are in boldface denote the tensor product of multiple subsystems, e.g., $|\boldsymbol{\emptyset}\rangle\langle\boldsymbol{\emptyset}|_{K_1K_2\ldots K_n} = \bigotimes_{t=1}^{n} |\emptyset\rangle\langle\emptyset|_{K_t}$.

(b) If Δ indicates an event, then Δ^c indicates the complementary event and $\Pr[\Delta]$ denotes the probability that event Δ occurs.

(c) We use the running indices i and $\vec{j}$ to indicate the sender and the set of receivers, respectively, for different protocol instances. Both indices run in the set of all parties: $\{i,\vec{j}\} \subset \{1,\ldots,n\}$. We refer to the sender and receivers ($\{i,\vec{j}\}$) as the *participants* of the protocol, while the remaining parties ($\{1,\ldots,n\} \setminus \{i,\vec{j}\}$) are called *nonparticipants*. In the case of multiple candidate senders, we indicate them with $\vec{i}$. Note that the size of the vectors $\vec{i}$ and $\vec{j}$ is not fixed.

(d) With $\mathcal{D} \subset \{1,\ldots,n\}$ we indicate the set of dishonest parties that may collaborate with the eavesdropper Eve.

(e) We denote by ID (for Identity Designation) the subprotocol of any ACKA and fully ACKA protocol that either unambiguously assigns the identities of sender and receivers to the parties in the network or aborts. In the case of ACKA, every receiver is also informed about the identity of the other participants when ID does not abort.

(f) Let $p(\vec{i},\vec{j})$ be the probability that, when running the ID protocol, the parties in the set $\vec{i}$ apply to become the sender and select their receivers in $\vec{j}$ (for instance, $\vec{j} = \{\vec{j}_{i_1},\vec{j}_{i_2},\ldots,\vec{j}_{i_{|\vec{i}|}}\}$, where $\vec{j}_{i_k}$ are the receivers selected by party i_k if they become the sender). If there is only one party applying to become the sender ($|\vec{i}| = 1$), then $\vec{j}$ is the set of receivers selected by the candidate sender i. The probability of this instance is indicated by $p(i,\vec{j})$.

(g) We define the following events.

(i) Γ: the ID subprotocol aborts from the point of view of every party in the network.
(ii) Φ: the ID subprotocol does not abort from the point of view of any party in the network and correctly designates the identity of the participants. That is, if party i is the only one applying to become the sender with intended receivers $\vec{j}$, then the protocol designates party i as the sender and $\vec{j}$ as the receivers. In the case of ACKA, the receivers are correctly informed about the identities of the other participants.
(iii) $\Omega_{\mathcal{P}}$: the ACKA (fully ACKA) protocol ends without aborting from the point of view of every participant.
(iv) $\Gamma_{\mathcal{P}}$: the ACKA (fully ACKA) protocol aborts from the point of view of every participant.

(h) Every party $t \in \{1,\dots,n\}$ holds three personal classical registers: P_t, K_t, and C_t.

(i) The register P_t stores information about the identity of party t and eventual information on the identity of the other participants, as assigned by the ID subprotocol. More specifically, $P_t = r$ if t is a receiver of a fully ACKA protocol, $P_t = r,i,\vec{j}$ if t is a receiver of an ACKA protocol, $P_t = s,\vec{j}$ if t is the sender with intended receivers in $\vec{j}$, and $P_t = \perp$ if t is a nonparticipant. We emphasize that r, s, and $\perp$ are just symbols used to discriminate the identity of the parties. Finally, $P_t = \emptyset$ if the protocol aborts during ID.
(ii) The register K_t stores either the conference key if t is a participant, or any information that the party might use to compute a guess of the conference key once the protocol is over if t is not a participant (a dishonest or honest-but-curious nonparticipant might want to learn the conference key). If the protocol aborts for party t, we set $K_t = \emptyset$.
(iii) The register C_t stores all the classical side information held by party t at any point in the protocol, which includes their private inputs and outputs to classical protocols and the public outputs.

(i) We indicate with K the set of registers K_t of every party, i.e., $K = K_1K_2\dots K_n$, and similarly for C and P. When we insert subscripts, it means that we restrict to the registers of the parties in the subscript. For instance, if $\mathcal{G} \subset \{1,\dots,n\}$ is a subset of parties, $K_{\mathcal{G}}$ is the set of conference key registers of the parties in $\mathcal{G}$. Similarly, $(PKC)_{\mathcal{G}}$ indicates the content of the registers P_t, K_t, and C_t for every party $t \in \mathcal{G}$. Finally, with the superscript ^{c} in $K^c_{\mathcal{G}}$, we denote the registers of the complement of $\mathcal{G}$.
(j) E is the quantum register of Eve.
(k) If the state of some registers depends on the value of a random variable X, we can express it as $\rho = \sum_x \Pr[X = x]\rho_{|x}$, where $\rho_{|x}$ is the state of the registers when $X = x$.
Let Δ be an event for the variable X, i.e., $\Pr[\Delta] = \sum_{x\in\Delta} \Pr[X = x]$. With $\rho_{|\Delta}$ we indicate the *normalized* state of the registers conditioned on the event Δ, $\rho_{|\Delta} := (1/\Pr[\Delta]) \sum_{x\in\Delta} \Pr[X = x]\rho_{|x}$. With $\rho_{\wedge\Delta}$ we indicate the *subnormalized* state conditioned on Δ whose trace corresponds to the probability of event Δ occurring: $\rho_{\wedge\Delta} := \sum_{x\in\Delta} \Pr[X = x]\rho_{|x}$.
(l) With $\rho_{PKCE|i,\vec{j}}$ ($\rho_{PKCE|\vec{i},\vec{j}}$) we indicate the state of registers P, K, C, and E conditioned on the event where party(ies) i ($\vec{i}$) applied to be the sender with the corresponding intended receivers in $\vec{j}$.
(m) If Alice is the sender and Bob_1, Bob_2, …, Bob_m are the m receivers, we replace the indices i and $\vec{j}$ pointing to their registers with A and $\vec{B}$, respectively. Thus, for instance, K_A is Alice's key register containing her conference key k_A and K_{B_l} is Bob_l's key register containing his conference key $\vec{k}_{B_l}$ (for $l \in \{1,\dots,m\}$).
(n) Let $|\vec{x}|$ be the number of entries of the string $\vec{x}$. Let $\omega_r(\vec{x})$ be the relative Hamming weight of the bitstring $\vec{x}$, i.e., $\omega_r(\vec{x}) := |\{k : x_k = 1\}| \,/\, |\vec{x}|$ '.
(o) The trace distance between two states ρ and σ is given by $\|\rho - \sigma\|_{\text{tr}} = \frac{1}{2}\text{Tr}[\sqrt{(\rho-\sigma)^2}]$ and is proportional to the trace norm of the operator $\rho - \sigma$.
(p) ρ^f_{PKCE} is the output state, or final state, of an ACKA (fully ACKA) protocol. We also simply indicate it as ρ^f.

APPENDIX B: SECURITY DEFINITIONS

As discussed in Sec. II, we identify three properties that an ideal ACKA (fully ACKA) protocol is expected to satisfy, namely, integrity, CKA-security, and anonymity. Here we introduce a formal security definition, which quantifies, for every property, how close the output state of the real protocol is from a state with the required property. Note that the output state ρ^f of a generic ACKA (fully ACKA) protocol can always be decomposed as the following:

$$\rho^f = \sum_{i,\vec{j}} p(i,\vec{j})\rho^f_{PKCE|i,\vec{j}} + \sum_{\vec{i},\vec{j}} p(\vec{i},\vec{j})\rho^f_{PKCE|\vec{i},\vec{j}}, \quad \text{(B1)}$$

where the first term contains the output states of the protocol when only one party applied to become the sender (party i), while the second term groups the output states

when multiple candidate senders are applied (parties $\vec{i}$). The output state of an integrous protocol is of the form

$$\sigma^{\text{in}} = \sum_{i,\vec{j}} p(i,\vec{j}) \Big(\Pr[\Phi|i,\vec{j}]\, \xi_P^{(i,\vec{j})} \otimes \rho^f_{KCE|i,\vec{j},\Phi} + \Pr[\Phi^c|i,\vec{j}]\, |\emptyset\rangle\langle\emptyset|_P \otimes \rho^f_{KCE|i,\vec{j},\Phi^c} \Big) + \sum_{\vec{i},\vec{j}} p(\vec{i},\vec{j})\, |\emptyset\rangle\langle\emptyset|_P \otimes \rho^f_{KCE|\vec{i},\vec{j}}, \tag{B2}$$

where $\xi_P^{(i,\vec{j})}$ is the ideal state of registers P, provided that party i applied to become the sender with receivers in $\vec{j}$ and the identities were correctly assigned. For an ACKA protocol the state $\xi_P^{(i,\vec{j})}$ reads

$$\xi_P^{(i,\vec{j})} := |s,\vec{j}\rangle\langle s,\vec{j}\,|_{P_i} \otimes |r,i,\vec{j}\rangle\langle r,i,\vec{j}\,|_{P_{\vec{j}}} \otimes |\perp\rangle\langle\perp|_{P^c_{i,\vec{j}}}, \tag{B3}$$

while for a fully ACKA protocol it reads

$$\xi_P^{(i,\vec{j})} := |s,\vec{j}\rangle\langle s,\vec{j}\,|_{P_i} \otimes |r\rangle\langle r|_{P_{\vec{j}}} \otimes |\perp\rangle\langle\perp|_{P^c_{i,\vec{j}}}. \tag{B4}$$

The distinction between the two states in Eqs. (B3) and (B4) is always clear from the context of the protocol being addressed (either an ACKA or a fully ACKA protocol).

We remark that the integrous state, Eq. (B2), is constrained only by the states of its P registers. As such, it is obtained from the output state of the real protocol (B1) by replacing the P register of every party with the abort symbol $\emptyset$, whenever event Φ^c occurs or when there are multiple candidate senders or no sender ($|\vec{i}| \neq 1$).

The output state of a CKA-secure protocol, conditioned on having honest participants ($\{i,\vec{j}\} \cap \mathcal{D} = \emptyset$) and on Φ, is of the form

$$\sigma^{\text{CKA}}_{KC^c_{i,\vec{j}}E|i,\vec{j},\Phi} = \Pr[\Omega_{\mathcal{P}}|i,\vec{j},\Phi]\, \tau_{K_{i,\vec{j}}} \otimes \rho^f_{(KC)^c_{i,\vec{j}}E|i,\vec{j},\Phi,\Omega_{\mathcal{P}}} + \Pr[\Omega^c_{\mathcal{P}}|i,\vec{j},\Phi]\, |\emptyset\rangle\langle\emptyset|_{K_{i,\vec{j}}} \otimes \rho^f_{(KC)^c_{i,\vec{j}}E|i,\vec{j},\Phi,\Omega^c_{\mathcal{P}}}, \tag{B5}$$

where the key registers $K_{i,\vec{j}}$ of the participants, for a key of ℓ bits, are perfectly correlated and random:

$$\tau_{K_{i,\vec{j}}} := \frac{1}{|\mathcal{K}|} \sum_{\vec{k}\in\mathcal{K}} |\vec{k}\rangle\langle\vec{k}|_{K_i} \otimes |\vec{\boldsymbol{k}}\rangle\langle\vec{\boldsymbol{k}}|_{K_{\vec{j}}}, \quad \mathcal{K} = \{0,1\}^\ell. \tag{B6}$$

Note that, similarly to Eq. (B2), the output state of a CKA-secure protocol (B5) is obtained by replacing the $K_{i,\vec{j}}$ registers in the state of the real protocol with the ideal outputs.

Finally, an ideal ACKA (fully ACKA) protocol must be anonymous, i.e., the identity of each participant must be kept secret from the nonparticipants and Eve—and from the other receivers in the case of fully ACKA. In this case we cannot simply replace the real output registers by their ideal counterparts, as in Eqs. (B2) and (B5), since anonymity is a property of the global quantum state accounting for different instances of sender and receivers. For example, if a protocol is anonymous with respect to a nonparticipant t, its output state satisfies $\rho^f_{P_tK_tC_t|i,\vec{j}} = \rho^f_{P_tK_tC_t|i',\vec{j}'}$ for $i \neq i'$ and $\vec{j} \neq \vec{j}'$, i.e., it is independent of the choice of sender and receivers. We remark that dishonest participants may broadcast their identity, or the identity of all the participants in the case of ACKA. Hence, in these scenarios we cannot require the output state to be independent of the identities that could be revealed.

We now introduce the formal definitions of anonymity for an ACKA and a fully ACKA protocol. We denote the output state of an anonymous ACKA (fully ACKA) protocol by $\sigma^{\mathcal{D}}$. This emphasizes the fact that the anonymity requirements on $\sigma^{\mathcal{D}}$ crucially depend on the set of dishonest parties $\mathcal{D}$. For instance, no anonymity requirement is imposed on the output state of an ACKA protocol for the instances in which some participant is dishonest.

Definition 4 (ACKA anonymity): Let $\mathcal{D}$ be the set of dishonest parties taking part in an ACKA protocol, with output state $\sigma^{\mathcal{D}}$ of the form (B1). Then the ACKA protocol is anonymous if for any subset of parties $\mathcal{G} \subseteq \{1,\ldots,n\}$ it holds that

$$\sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|i,\vec{j}} = \sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|i',\vec{j}'} \quad \forall\, i,i',\vec{j},\vec{j}' \notin \mathcal{G}\cup\mathcal{D}, \tag{B7}$$

$$\sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|\vec{i},\vec{j}} = \sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|\vec{i}',\vec{j}'} \quad \forall\, \vec{i},\vec{i}',\vec{j},\vec{j}' \notin \mathcal{G}\cup\mathcal{D}, \tag{B8}$$

where $\sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|i,\vec{j}} = \text{Tr}_{P^c_{\mathcal{G}}K^c_{\mathcal{G}}C^c_{\mathcal{G}}}[\sigma^{\mathcal{D}}_{PKCE|i,\vec{j}}]$.

The anonymity conditions (B7) and (B8) establish that the final state of any subset of nonparticipants and Eve is independent of the identity of the remaining parties. In other words, the reduced output states $\sigma^{\mathcal{D}}_{(PKC)^c_{i,\vec{j}}E|i,\vec{j}}$ and $\sigma^{\mathcal{D}}_{(PKC)^c_{i,\vec{j}}E|\vec{i},\vec{j}}$ do not contain any information about the identities of the participants. Note that if a participant would be contained in $\mathcal{G}$, the register $P_{\mathcal{G}}$ would also carry information about the identities of all the other participants (recall that an ACKA protocol reveals the participants' identities to each participant) and therefore there would be no condition to be satisfied. Moreover, note that Eqs. (B7) and (B8) do not impose any condition if the sender(s) or receivers are dishonest, since nothing prevents them from broadcasting the identities of all the participants.

Definition 5 (Fully ACKA anonymity): Let $\mathcal{D}$ be the set of dishonest parties taking part in a fully ACKA protocol, with output state $\sigma^{\mathcal{D}}$ of the form (B1). Then the fully ACKA protocol is anonymous if for any subset of parties $\mathcal{G} \subseteq \{1,\dots,n\}$ it holds that

$$\sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|i,\vec{j}} = \sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|i',\vec{j}'} \forall\, i,i',\vec{j},\vec{j}' : i,i' \notin \mathcal{G}\cup\mathcal{D},\ \vec{j}\cap\mathcal{G} = \vec{j}'\cap\mathcal{G}, \vec{j}\cap\mathcal{D} = \vec{j}'\cap\mathcal{D}, \tag{B9}$$

$$\sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|\vec{i},\vec{j}} = \sigma^{\mathcal{D}}_{P_{\mathcal{G}}K_{\mathcal{G}}C_{\mathcal{G}}E|\vec{i}',\vec{j}'} \forall\, \vec{i},\vec{i}',\vec{j},\vec{j}' : \vec{i},\vec{i}' \notin \mathcal{G}\cup\mathcal{D},\ \vec{j}\cap\mathcal{G} = \vec{j}'\cap\mathcal{G}, \vec{j}\cap\mathcal{D} = \vec{j}'\cap\mathcal{D}. \tag{B10}$$

The anonymity conditions (B9) and (B10) extend the requirements of Eqs. (B7) and (B8) due to the fact that in a fully ACKA protocol the receiver is unaware of the identity of the other participants. Indeed, conditions (B9) and (B10) establish that in all the instances of the protocol in which the parties in subsets $\mathcal{G}$ and $\mathcal{D}$ have a fixed role (except for the role of sender), their reduced state is independent of the identities of the other parties.

Definition 6, which is a formal restatement of Definition 1, defines the security of an ACKA (fully ACKA) protocol through the trace distances of the output state of the protocol from an integrous state (B2), a CKA-secure state (B5), and an anonymous state satisfying Definition 4 (Definition 5), respectively.

Definition 6 (Security (rigorous)): An ACKA (fully ACKA) protocol with dishonest parties in $\mathcal{D}$ is ε-secure, with $\varepsilon = \varepsilon_{\text{in}} + \varepsilon_{\text{CKA}} + \varepsilon_{\text{an}}$, if it satisfies the following three conditions.

(a) ε_{in}-integrity:

$$\max_{i,\vec{j}} \Pr[\Gamma^c \cap \Phi^c | i,\vec{j}] \le \varepsilon_{\text{in}} \ \wedge\ \max_{\vec{i},\vec{j}} \Pr[\Gamma^c|\vec{i},\vec{j}] \le \varepsilon_{\text{in}}, \tag{B11}$$

(b) ε_{CKA}-CKA-security:

$$\max_{\{i,\vec{j}\}\cap\mathcal{D}=\emptyset} \Pr[\Omega_{\mathcal{P}}|i,\vec{j},\Phi] \left\| \rho^{f}_{KC^c_{i,\vec{j}}E|i,\vec{j},\Phi,\Omega_{\mathcal{P}}} - \tau_{K_{i,\vec{j}}} \otimes \rho^{f}_{(KC)^c_{i,\vec{j}}E|i,\vec{j},\Phi,\Omega_{\mathcal{P}}} \right\|_{\text{tr}} + \Pr[\Omega^c_{\mathcal{P}} \cap \Gamma^c_{\mathcal{P}} | i,\vec{j},\Phi] \le \varepsilon_{\text{CKA}}, \tag{B12}$$

(c) ε_{an}-anonymity:

$$\max_{i,\vec{j}} \left\| \rho^{f}_{(PKC)^c_{i,\vec{j}}E|i,\vec{j}} - \sigma^{\mathcal{D}}_{(PKC)^c_{i,\vec{j}}E|i,\vec{j}} \right\|_{\text{tr}} \le \varepsilon_{\text{an}} \ \wedge\ \max_{\vec{i},\vec{j}} \left\| \rho^{f}_{(PKC)^c_{\vec{i},\vec{j}}E|\vec{i},\vec{j}} - \sigma^{\mathcal{D}}_{(PKC)^c_{\vec{i},\vec{j}}E|\vec{i},\vec{j}} \right\|_{\text{tr}} \le \varepsilon_{\text{an}} \quad \text{(ACKA)}, \tag{B13}$$

$$\max_{i,\vec{j}} \left\| \rho^{f}_{(PKC)^c_{i}E|i,\vec{j}} - \sigma^{\mathcal{D}}_{(PKC)^c_{i}E|i,\vec{j}} \right\|_{\text{tr}} \le \varepsilon_{\text{an}} \ \wedge\ \max_{\vec{i},\vec{j}} \left\| \rho^{f}_{(PKC)^c_{\vec{i}}E|\vec{i},\vec{j}} - \sigma^{\mathcal{D}}_{(PKC)^c_{\vec{i}}E|\vec{i},\vec{j}} \right\|_{\text{tr}} \le \varepsilon_{\text{an}} \quad \text{(fully ACKA)}, \tag{B14}$$

where ρ^f is the output state of the ACKA (fully ACKA) protocol, $\sigma^{\mathcal{D}}$ is the output state of any ACKA (fully ACKA) protocol, which satisfies Definition 4 (Definition 5), and $\tau_{K_{i,\vec{j}}}$ is given in Eq. (B6).

We remark that to simplify the integrity condition, we replaced the trace distance from the integrous state (B2) with a sufficient condition on the probabilities of unwanted events.

The CKA-security condition (B12) establishes that the final state of the protocol, conditioned on honest parties i and $\vec{j}$ correctly designated as sender and receivers, is close to a state in which an ideal key (B6) is distributed to the participants or where the protocol aborts for every participant. Indeed, in the ACKA scenario, it could happen that the protocol aborts for some participants but not for others (event $\Omega^c_{\mathcal{P}} \cap \Gamma^c_{\mathcal{P}}$). This would spoil the CKA-security of the protocol, since not all the participants would end up with the same conference key (recall that if the protocol aborts for party t, then $K_t = \emptyset$). We remark that in the standard QKD and CKA scenarios there is no need to account for such instances, since the identities of the participants are public and they can communicate over the authenticated classical channel, thus agreeing on when the protocol aborts.

Finally, the anonymity conditions (B13) and (B14) account for deviations of the real protocol from an anonymous protocol, as defined by Definitions 4 and 5, respectively. Nevertheless, since the property of anonymity is intertwined with integrity and CKA-security,

any issue of the real protocol with regards to the latter may at the same time increase its deviation from a perfectly anonymous protocol.

We observe that the conditions (B11)–(B14) are independent of the distribution $\{p(i,\vec{j}),p(\vec{i},\vec{j})\}$, which describes the probability that certain parties apply to become senders with specified receivers. This is a desirable feature since the distribution may not be accessible to a party who wants to prove the protocol's security.

As discussed in Sec. II, our fully ACKA protocol (Protocol 2) does not satisfy the anonymity condition (B14) of Definition 6, but rather a weaker anonymity condition provided in Definition 2 and formally stated here.

Definition 7 (Weak anonymity for fully ACKA (rigorous)): Let $\mathcal{D}$ be the set of dishonest parties taking part in a fully ACKA protocol, with output state ρ^f. The fully ACKA protocol is $\varepsilon_{W\text{an}}$-weak-anonymous, with $\varepsilon_{\text{Wan}} = \varepsilon_{\text{NPan}} + \varepsilon_{\text{Pan}}$, if the following two conditions are satisfied:

1. (Anonymity with respect to nonparticipants and Eve.) The output state of the protocol satisfies the anonymity condition (B13) for ACKA protocols:

$$\max_{i,\vec{j}} \left\| \rho^f_{(PKC)^c_{i,\vec{j}}E|i,\vec{j}} - \sigma^{\mathcal{D}}_{(PKC)^c_{i,\vec{j}}E|i,\vec{j}} \right\|_{\text{tr}} \leq \varepsilon_{\text{NPan}} \wedge$$
$$\max_{\vec{i},\vec{j}} \left\| \rho^f_{(PKC)^c_{\vec{i},\vec{j}}E|\vec{i},\vec{j}} - \sigma^{\mathcal{D}}_{(PKC)^c_{\vec{i},\vec{j}}E} \right\|_{\text{tr}} \leq \varepsilon_{\text{NPan}}, \tag{B15}$$

where $\sigma^{\mathcal{D}}$ satisfies Definition 4.

2. (Anonymity with respect to honest-but-curious receivers.) When the protocol's specifications are invariant under permutations of parties, any subset of honest-but-curious receivers $\mathcal{R} \subseteq \vec{j}$ cannot guess the identity of other participants with a higher probability than the trivial guess, except for a small deviation ε_{Pan}. In formulas, the probability of correctly guessing the set of participants is bounded as follows:

$$p_{\text{guess}} \leq \max_{i,\vec{j}\supseteq\mathcal{R}} p_{\mathcal{R}}(i,\vec{j}) + \varepsilon_{\text{Pan}}, \tag{B16}$$

$$p_{\text{guess}} \leq \max_{\vec{i},\vec{j}\supseteq\mathcal{R}} p_{\mathcal{R}}(\vec{i},\vec{j}) + \varepsilon_{\text{Pan}}, \tag{B17}$$

where $p_{\mathcal{R}}(i,\vec{j})$ $(p_{\mathcal{R}}(\vec{i},\vec{j}))$ is defined as $p_{\mathcal{R}}(i,\vec{j}) := p(i,\vec{j})/(\sum_{i,\vec{j}\supseteq\mathcal{R}} p(i,\vec{j}))$ $(p_{\mathcal{R}}(\vec{i},\vec{j}) := p(\vec{i},\vec{j})/(\sum_{\vec{i},\vec{j}\supseteq\mathcal{R}} p(\vec{i},\vec{j})))$.

APPENDIX C: SUBPROTOCOLS

Before presenting the ACKA and fully ACKA protocols, we introduce the subprotocols on which they built upon. Such protocols involve many different bitstrings, whose length in many cases is not defined to be an integer number, unless a ceiling or floor function is applied. In order not to increase the complexity of the notation, we omit the ceiling and floor functions. Note that, from the point of view of the plots, this omission is irrelevant since few bits of difference do not sensibly modify the key rates (consider that all plots start from $L_{\text{tot}} = 10^5$).

To start with, we make use of the classical Parity, Veto, and Collision Detection protocols from [20]. The first two protocols compute the Parity and the logical OR of their inputs, respectively, while Collision Detection detects the presence of multiple parties applying to become the sender. Importantly, in our version of the Parity protocol we do not require simultaneous broadcast, contrary to Ref. [20].

Protocol 3: Parity [20]

Let $x_t \in \{0,1\}$ be the input of party t and y_t the output of the protocol for party t. Then $y_t = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ for every t.

Every party $t \in \{1,\ldots,n\}$ does the following.

1. Select uniformly at random an n-bit string $\vec{r}_t = r_t^1 r_t^2 \ldots r_t^n$ such that $x_t = \oplus_{j=1}^n r_t^j$.
2. Send r_t^j to party j and keep r_t^t.
3. Compute $z_t = \oplus_{j=1}^n r_j^t$, i.e., the Parity of the bits received (including r_t^t).
4. Broadcast z_t.
5. Compute $y_t = \oplus_{k=1}^n z_k$ to obtain the Parity of the inputs $\{x_t\}_t$.

Protocol 4: Veto [20]

Let $x_t \in \{0,1\}$ be the input of party t and y_t the output of the protocol for party t. Then $y_t = x_1 \vee x_2 \vee \cdots \vee x_n$ for every $t \in \{1,\ldots,n\}$.

1. Initialize $y_t = 0\ \forall\, t$.
2. For every party t, repeat the following r_V times:

2.1. Each party j sets the value of q_j according to the following:

$$\begin{array}{ll} q_j = 0 & \text{if } x_j = 0 \\ q_j \in_R \{0,1\} & \text{if } x_j = 1\text{'} \end{array} \tag{C1}$$

where $q_j \in_R \{0,1\}$ denotes that q_j is picked uniformly at random in $\{0,1\}$.

2.2. The parties execute the Parity protocol (Protocol 3) with inputs $q_1, q_2, \ldots, q_n$, such that party t is the last to broadcast. If the outcome of the Parity protocol is 1 or any party refuses to broadcast, then set $y_t = 1\ \forall\, t \in \{1,\ldots,n\}$.

Note that, due to the probabilistic action in step 2.1, Protocol 4 allows a given party t to learn whether any other party has input 1 even when the input of party t is $x_t = 1$, which would not happen in an ideal implementation of the Veto function [20]. This crucial fact allows a candidate

sender to detect the presence of other candidate senders in the first Veto performed in Collision Detection.

Protocol 5: Collision Detection [20]

This protocol is used to detect the presence of multiple candidate senders, or no senders at all. Let $x_t \in \{0,1,2\}$ be the input of party t and y_t the output of the protocol for party t. Then the output of every party $t \in \{1,\dots,n\}$ is interpreted as follows:

$$\begin{array}{ll} y_t = 0 & \text{no sender applies and the protocol aborts} \\ y_t = 1 & \text{one sender applied and the protocol proceeds} \\ y_t = 2 & \text{a collision is detected (multiple senders)} \\ & \text{and the protocol aborts} \end{array} \tag{C2}$$

The protocol steps are given in the following.

1. Veto A:
   1.1. Each party j sets $a_j = \min\{x_j, 1\}$.
   1.2. All participants perform the Veto protocol (Protocol 4) with input $a_1, a_2, \dots, a_n$.
2. Veto B (skip if Veto A outputs 0):
   2.1. Each party j sets

$$\begin{array}{ll} b_j = 1 & \text{if } x_j = 2, \text{ or if } x_j = 1 \text{ and party } j \\ & \text{detected that another party had input 1} \\ & \text{in Veto A} \\ b_j = 0 & \text{otherwise} \end{array} \tag{C3}$$

   2.2. All participants perform the Veto protocol (Protocol 4) with input $b_1, b_2, \dots, b_n$.
3. Every party $t \in \{1,\dots,n\}$ sets their output to

$$\begin{array}{ll} y_t = 0 & \text{if the output of Veto A is 0} \\ y_t = 1 & \text{if the outputs of Veto A and B sum to 1} \\ y_t = 2 & \text{if the outputs of Veto A and B sum to 2} \end{array} \tag{C4}$$

In the Collision Detection protocol (Protocol 5), the input $x_t = 2$ can describe the action of a dishonest party that wants to force the protocol to detect a collision and output $y_t = 2$.

In the ACKA protocol, Alice must disclose her identity and the identities of the other Bobs to every Bob. This is achieved by transmitting a bitstring containing the positions of the sender and the chosen receivers in the network. In particular, let $\vec{d}_t$ be an $(n - 1 + \log_2 n)$-bit long string that Alice transmits to every other party t. If Alice is transmitting d_t to a Bob, then the first bit of d_t is 1, the following $\log_2 n$ bits contain the position of Alice with respect to some predefined ordering of the n parties, and the remaining $n - 2$ bits indicate the identities of the other parties (bit 1 for receiver and bit 0 for a nonparticipant), except for Alice and the Bob receiving d_t. If instead Alice is transmitting d_t to a nonparticipant, then the first bit is 0 and the remaining bits are random.

In order to ensure that the transmission of $\vec{d}_t$ is error-free, Alice encodes the bitstring with a public encoding algorithm (F, G) composed of an encoding function F and a decoding function G, denoted Algebraic Manipulation Detection (AMD) code [20,39]. An AMD code has the important property of detecting, with high probability, any tampering with the encrypted message, as stated in the following lemma.

Lemma 8 (AMD code [20,39]): Let $\vec{x}$ be a bitstring. There exists an AMD code (F, G), whose encoded string $F(\vec{x})$ has length

$$|F(\vec{x})| = |\vec{x}| + 2\left(\log_2 |\vec{x}| + \log_2 \frac{1}{\varepsilon_{\text{enc}}}\right), \tag{C5}$$

such that $G(F(\vec{x})) = \vec{x}$ for every $\vec{x}$ and such that $\Pr[G(F(\vec{x}) \oplus \vec{b}) \neq \top] \leq \varepsilon_{\text{enc}}$ for every bitstring $\vec{b} \neq \vec{0}$ chosen without prior knowledge of the encoded bitstring $F(\vec{x})$, while a complete knowledge of $\vec{x}$ is allowed [39] [note that, due to the probabilistic nature of an AMD code, $F(\vec{x})$ can still be unknown even when the input $\vec{x}$ is known].

For the use in ACKA, the encoded string $F(\vec{d}_t)$ has length

$$|F(\vec{d}_t)| = n - 1 + \log_2 n + 2\left(\log_2(n - 1 + \log_2 n) + \log_2 \frac{1}{\varepsilon_{\text{enc}}}\right), \tag{C6}$$

and it is then used by Alice as her input in a sequence of Parity protocols in order to anonymously transmit it to party t. This is the core of the identity designation subprotocol used by the ACKA protocol, named ACKAID.

Protocol 6: Identity designation for ACKA (**ACKAID**)

Before the protocol starts, the n parties set $v_t = 0\ \forall t$.

1. The n parties perform the Collision Detection protocol (Protocol 5), where party t inputs 1 if they want to be the sender or 0 otherwise. If the Collision Detection outputs 0 or 2 then the ACKA protocol aborts for all the n parties. Else, they proceed with the next steps.
2. If the protocol does not abort in step 1, then there is a single sender, whom we identify as Alice, except for a small probability.
3. For every party $t \in \{1,\dots,n\}$, do the following.
   3.1. The n parties perform the Parity protocol for $|F(\vec{d}_t)|$ times with the following inputs. If $t \neq A$,

Alice uses $F(\vec{d}_t)$, party t uses a random bitstring $\vec{r}_t$ and the other parties input $\vec{0}$. If $t = A$, then Alice uses $F(\vec{0}) \oplus \vec{r}$, where $\vec{r}$ is a random bitstring, and the other parties input $\vec{0}$. Let $\vec{o}$ be the output of the Parity protocols.

3.2. Party t computes $G(\vec{r}_t \oplus \vec{o})$. If G returns $\top$, party t sets $v_t = 1$, otherwise party t recovers their identity as assigned by Alice (and eventually the identity of the other participants if they are a receiver). If $t = A$, then Alice computes $G(\vec{r} \oplus \vec{o})$. If G returns $\top$, Alice sets $v_t = 1$.

4. The n parties perform the Veto protocol (Protocol 4) with inputs v_t. If the Veto outputs 1, the ACKA protocol aborts for all the n parties.

The total number of bipartite channel uses required by the ACKAID protocol is $n^2(n-1)(3r_V + |F(\vec{d}_t)|)$, where $|F(\vec{d}_t)|$ is given by Eq. (C6).

Recall that, differently from ACKA, in a fully ACKA protocol Alice needs only to communicate to each party whether they are a Bob or a nonparticipant. Thus, we present the fully ACKAID protocol, where we denote d_t the bit that Alice sends to every other party t to communicate their identity. In particular, $d_t = 1$ ($d_t = 0$) means that party t is a receiver (nonparticipant). In order to detect bit flips in the transmission of d_t, Alice encodes d_t with an AMD code [20,39] (F, G), which satisfies the statement of Lemma 8 and is such that

$$|F(d_t)| = 1 + 2\log_2 \frac{1}{\varepsilon_{\text{enc}}}. \tag{C7}$$

Protocol 7: Identity Designation for fully ACKA (**fully ACKAID**)

Before the protocol starts, the n parties set $v_t = 0\ \forall t$.

1. The n parties perform the Collision Detection protocol (Protocol 5), where party t inputs 1 if they want to be the sender or 0 otherwise. If the Collision Detection outputs 0 or 2 then the fully ACKA protocol aborts for all the n parties. Else, they proceed with the next steps.
2. If the protocol does not abort in step 1, then there is a single sender, whom we identify as Alice, except for a small probability.
3. For every party $t \in \{1, \ldots, n\}$, do the following.

3.1. The n parties perform the Parity protocol for $|F(d_t)|$ times with the following inputs. If $t \neq A$, Alice uses $F(d_t)$, party t uses a random bitstring $\vec{r}_t$ and the other parties input $\vec{0}$. If $t = A$, then Alice uses $F(0) \oplus \vec{r}$, where $\vec{r}$ is a random bitstring, and the other parties input $\vec{0}$. Let $\vec{o}$ be the output of the Parity protocols.

3.2. Party t computes $G(\vec{r}_t \oplus \vec{o})$. If G returns $\top$, party t sets $v_t = 1$, otherwise party t recovers their identity as assigned by Alice. If $t = A$, then Alice computes $G(\vec{r} \oplus \vec{o})$ and if G returns $\top$, Alice sets $v_t = 1$.

4. The n parties perform the Veto protocol (Protocol 4) with inputs v_t. If the Veto outputs 1, the fully ACKA protocol aborts for all the n parties.

The total number of bipartite channel uses required by the fully ACKAID protocol is $n^2(n-1)(3r_V + |F(d_t)|)$, where $|F(d_t)|$ is given by Eq. (C7).

As discussed in Sec. II, the fully ACKA scenario prevents Alice from distributing the testing key through a pre-established conference key shared by all participants. For this, we introduce another subprotocol called the Testing Key Distribution protocol. With this protocol, Alice anonymously provides each Bob_l (for $l \in \{1, \ldots, m\}$) with a key $\vec{k}_l$ that is given by the concatenation of two independent bitstrings: $\vec{k}_l = (\vec{k}_T, \vec{r}_l)$, where $\vec{k}_T$ ($|\vec{k}_T| = Lh(p)$) is the testing key and $\vec{r}_l$ is additional randomness used in the error-correction phase of fully ACKA. The string $\vec{r}_l$ is obtained by encoding the concatenated random string $(b_l, \vec{r}_\emptyset)$ with an AMD code (F, G): $\vec{r}_l = F(b_l, \vec{r}_\emptyset)$. In principle, Bob_l can recover the concatenated string by computing $G(\vec{r}_l) = (b_l, \vec{r}_\emptyset)$ and use it in error correction. In particular, the random bit b_l and the random string $\vec{r}_\emptyset$, with $|\vec{r}_\emptyset| = 1 + 2\log_2 1/\varepsilon_{\text{enc}}$, are employed in two separate steps of error correction. By adding the lengths of the two bitstrings composing $\vec{k}_l$, we obtain the total length of $\vec{k}_l$:

$$|\vec{k}_l| = Lh(p) + 4\left(1 + \log_2 \frac{1}{\varepsilon_{\text{enc}}}\right) + 2\log_2\left(1 + \log_2 \frac{1}{\varepsilon_{\text{enc}}}\right). \tag{C8}$$

In order not to reveal the number m of Bobs chosen by Alice, the distribution of the key $\vec{k}_l$ is repeated for $n-1$ times. In each iteration, Alice first notifies one party, say party s, to be the recipient of the key $\vec{k}_s = (\vec{k}_T, \vec{r}_s)$ and then runs a sequence of Parity protocols to transmit the bits of the key, with $s \neq A$. In case the recipient is not a Bob, Alice sums modulo two the bits of the key with random bits before transmitting them. Alice's notification procedure is inspired by the Notification protocol in Ref. [20] and it successfully notifies party s with probability at least $1 - 2^{-r_N}$, thanks to the iteration of r_N rounds.

Protocol 8: Testing Key Distribution (**TKD**)

Let $y_t \in \{0, 1\}$ be the output of the Notification subprotocol for party t: if $y_t = 1$, party t has been notified as the

recipient. Before the TKD protocol starts, we set the verification bits of Alice and Bob$_l$ to $v_A = v_{B_l} = 0$ for every $l \in \{1, \ldots, m\}$. The following sequence of steps is repeated $n-1$ times.

1. Alice randomly picks a party $s \in \{1, \ldots, n\}$ that has not been notified in previous iterations.
2. **Notification:** the n parties repeat the following steps for every party $t \in \{1, \ldots, n\}$.

   2.1. Initialize $y_t = 0$.
   2.2. Repeat r_N times.

      2.2.1. Every party $j \neq A$ sets $p_j = 0$. Alice sets

$$\begin{array}{ll} p_A = 0 & \text{if } t \neq s \\ p_A \in_R \{0,1\} & \text{if } t = s. \end{array} \tag{C9}$$

      2.2.2. The n parties perform Parity (Protocol 3) with inputs $p_1, p_2, \ldots, p_n$ but party t does not broadcast. In this way party t is the only who can compute the outcome of the Parity protocol. If the Parity outcome is 1, party t sets $y_t = 1$.

3. The party s is notified ($y_s = 1$) with high probability.
4. **Distribution:** the n parties perform the Parity protocol for $|\vec{k}_l|$ times with the following inputs. If party s corresponds to Bob$_l$ (for $l \in \{1, \ldots, m\}$), Alice inputs $\vec{k}_l$, Bob$_l$ inputs a random bitstring $\vec{r}$, and the other parties input $\vec{0}$. Otherwise, if s is a nonparticipant, Alice inputs $\vec{k}_s \oplus \vec{r}$, where $\vec{r}$ is a random string, and the other parties input $\vec{0}$. Let $\vec{o}$ be the output of the Parity protocols.
5. If s corresponds to Bob$_l$, he computes $\vec{o} \oplus \vec{r}$ and recovers the testing key k_T. Moreover, Bob$_l$ applies the decoding function G on the last $|\vec{r}_l|$ bits of $\vec{o} \oplus \vec{r}$. If G returns $\top$, then Bob$_l$ sets $v_{B_l} = 1$. Otherwise, Bob$_l$ recovers the bit b_l and the string $\vec{r}_\emptyset$ to be used in error correction.If s is a nonparticipant, Alice computes $\vec{o} \oplus \vec{r}$ and applies the decoding function G on the last $|\vec{r}_s|$ bits of $\vec{o} \oplus \vec{r}$. If G returns $\top$, then Alice sets $v_A = 1$.

If Bob$_l$ (for $l \in \{1, \ldots, m\}$) has not been notified in none of the $n-1$ notification rounds, he sets $v_{B_l} = 1$.

The total number of bipartite channel uses required for the TKD protocol is $n^2(n-1)^2 r_N + n(n-1)^2|\vec{k}_l|$, where $|\vec{k}_l|$ is given in Eq. (C8).

The verification bits v_A and v_{B_l} are used to abort the fully ACKA protocol in step 7 (cf. Protocol 2) if a dishonest party attempts to modify the random bit b_l or the random string $\vec{r}_\emptyset$ destined to Bob$_l$. The fact that Alice and the Bobs verify this fact in the exact same way, prevents nonparticipants and Eve from learning the identity of the participant who causes fully ACKA to abort.

Additionally, the bits v_{B_l} are used to abort the fully ACKA protocol in step 7 if some Bob has not been notified and has not received $\vec{k}_l$. Note that, in order not to reveal the number of Bobs from the outcome of step 7, we will require the nonparticipants to behave like a Bob if they have not been notified in TKD.

Finally, we illustrate the one-way error-correction protocols adopted in ACKA (Protocol 9) and fully ACKA (Protocol 10), where Alice provides the Bobs with a syndrome that allows them to correct their faulty raw keys and match Alice's. Together with the syndrome, Alice distributes a hash of her raw key so that each Bob can verify the success of his error-correction procedure.

Protocol 9: Error correction for ACKA (**ACKAEC**)

1. Alice computes the syndrome $\vec{y}$, with $|\vec{y}| = L(1-p)h(Q_Z)$, from her raw key, i.e., from the string of measurement outcomes corresponding to key generation rounds in $\vec{k}'_T$.
2. All the parties broadcast a random string of $|\vec{y}|$ bits, while Alice broadcasts $\vec{y} \oplus \vec{k}_2$, where $\vec{k}_2$ is extracted from a previously established conference key.
3. From the knowledge of $\vec{k}_2$, each Bob recovers $\vec{y}$ from Alice's broadcast and uses it to correct his raw key.
4. In order to verify if the error correction is successful, the public randomness picks a two-universal hash function mapping keys of $L(1-p)$ bits to keys of $b_h := \log_2 \frac{n-1}{\varepsilon_{\mathrm{EC}}}$ bits. Alice and the Bobs compute the hashes $\vec{h}_A$ and $\vec{h}_{B_l}$ (for $l \in \{1, \ldots, m\}$) by applying the hash function on their (error-corrected) raw keys.
5. All the parties broadcast a b_h-bit random string, except for Alice who broadcasts $\vec{h}_A \oplus \vec{k}_3$, where $\vec{k}_3$ is extracted from a previously established conference key.
6. Each Bob recovers $\vec{h}_A$ and compares it with his hash $\vec{h}_{B_l}$. If $\vec{h}_A \neq \vec{h}_{B_l}$, then the error-correction procedure failed for Bob$_l$ and he sets $v_{B_l} = 1$, otherwise he sets $v_{B_l} = 0$.
7. All the parties broadcast a bit. Alice (Bob$_l$, for $l \in \{1, \ldots, m\}$) broadcasts $v_A \oplus b_A$ ($v_{B_l} \oplus b_l$), where b_A (b_l) is extracted from a previously established conference key. The predefined ordering of the n parties could be used to assign the bits b_A and b_l of a previous key to Alice and Bob$_l$, respectively.
8. From the knowledge of the bits b_A and b_l, Alice and the Bobs learn if the verification bit v_A or v_{B_l} of any participant is equal to 1, in which case they consider the protocol aborted and set their conference keys to $\vec{k}_A = \vec{k}_{B_l} = \emptyset$ for $l \in \{1, \ldots, m\}$.

Protocol 10: Error correction for fully ACKA (**fully ACKAEC**)

1. The public source of randomness picks a two-universal hash function. Alice uses it to compute the syndrome $\vec{y}$, an $L(1-p)h(Q_Z)$-bit long string, from her raw key, i.e., from the string of measurement outcomes corresponding to key generation rounds in $\vec{k}'_T$.
2. The n parties repeatedly perform the Parity protocol for $L(1-p)h(Q_Z)$ times. Alice uses as an input the bits of $\vec{y}$, while the other parties input $\vec{0}$. The output string of the Parity protocols is $\vec{o}_1$.
3. Each Bob uses $\vec{o}_1$ as the syndrome to correct his raw key [45,46].
4. In order to verify if the error correction is successful, the public randomness picks a two-universal hash function mapping keys of $L(1-p)$ bits to keys of $b_h := \log_2 \frac{n-1}{\varepsilon_{\mathrm{EC}}}$ bits. Alice and the Bobs compute the hashes $\vec{h}_A$ and $\vec{h}_{B_l}$ (for $l \in \{1,\ldots,m\}$) by applying the hash function on their (error-corrected) raw keys.
5. The n parties perform the Parity protocol b_h times, using input $\vec{0}$ except for Alice who uses the bits of $\vec{h}_A$. The output string of the Parity protocols is $\vec{o}_2$.
6. Each Bob compares $\vec{o}_2$ with his hash $\vec{h}_{B_l}$. If $\vec{o}_2 \neq \vec{h}_{B_l}$ then Bob$_l$ sets $v_{B_l} = 1$, otherwise he sets $v_{B_l} = 0$. Then, all the parties broadcast a random bit, except for Bob$_l$ who broadcasts $b_l \oplus v_{B_l}$ (for $l \in \{1,\ldots,m\}$).
7. From the knowledge of the bits b_l, Alice retrieves the verification bits v_{B_l} from the broadcast.
8. The n parties execute the Parity protocol $|\vec{r}_\emptyset|$ times, where every party except for Alice inputs $\vec{0}$. If $v_{B_l} = 0$ for every l and if $\vec{o}_2 = \vec{h}_A$, Alice inputs $\vec{r}_\emptyset \oplus F(0)$, otherwise she inputs $\vec{r}_\emptyset \oplus F(1)$. Let $\vec{o}_3$ be the output of the Parity protocols.
9. Alice considers the fully ACKA protocol aborted if $\vec{o}_3$ differs from her input or if she inputs $\vec{r}_\emptyset \oplus F(1)$, while every Bob computes $G(\vec{r}_\emptyset \oplus \vec{o}_3)$. If $G(\vec{r}_\emptyset \oplus \vec{o}_3) \in \{\top, 1\}$, the Bobs consider the fully ACKA protocol aborted.

Note that if a dishonest party flipped some of the bits of $\vec{o}_1$ or $\vec{o}_2$ or $\vec{o}_3$ by inputting 1 in the Parity protocols, then the fully ACKA protocol aborts with high probability for Alice and all the Bobs.

APPENDIX D: PROTOCOLS WITH GHZ STATES

Here we provide a more detailed description of Protocol 1 (ACKA) and Protocol 2 (fully ACKA), which anonymously extract a conference key thanks to the multipartite entanglement of GHZ states. The protocols' parameters are summarized in Table I. We prove the protocols' security in the Supplemental Material [38].

Protocol 1: Anonymous conference key agreement (**ACKA**)

1. The parties perform the ACKAID protocol (Protocol 6). If the ACKAID protocol does not abort, Alice is guaranteed to be the only sender and the Bobs learn the identities of Alice and of each other, except for a small probability.
2. Alice and the Bobs recover a shared conference key previously established.
3. Alice generates a random bitstring of length L where 1 corresponds to a test round and 0 to a key generation round. Given that p is the probability that a round is identified as a test round, she compresses the string to a testing key $\vec{k}_T$ of length $Lh(p)$ [typically $h(p) < 8\%$]. All the parties broadcast a random string of $Lh(p)$ bits, except for Alice who broadcasts $\vec{k}_T \oplus \vec{k}_1$, where $\vec{k}_1$ is extracted from a previously established conference key. Thanks to the knowledge of $\vec{k}_1$, each Bob recovers the testing key $\vec{k}_T$ from Alice's broadcast.
4. Repeat the following for L rounds.
   - 4.1. An n-party GHZ state is distributed to the n parties.
   - 4.2. Alice and the Bobs measure their qubits according to the testing key $\vec{k}_T$. They measure in the Z basis if the round is a key generation round, or in the X basis if the round is a test round. All the other parties measure X.
5. Once all the qubits have been measured (bounded storage assumption), the testing key $\vec{k}_T$ is publicly revealed. This is done anonymously by iterating the Parity protocol (Protocol 3) $Lh(p)$ times. In each instance of the Parity protocol, Alice inputs a bit of $\vec{k}_T$, while the other parties input 0. The output of the Parity protocols is $\vec{k}'_T$.
6. For every round in step 3 that is labeled as a test round by $\vec{k}'_T$, the n parties perform the Parity protocol with the following inputs. Let X_t (for $t \in \{1,\ldots,n\}$) be the outcome of party t if they measured X in that round, otherwise $X_t \in_R \{0,1\}$. Every party except for Alice ($t \neq A$) inputs X_t, while Alice inputs $T_A \in_R \{0,1\}$. Let $\vec{o}_T$ be the output of the Parity protocols for all the test rounds in $\vec{k}'_T$. Alice computes $Q_X^{\mathrm{obs}} = \omega_r(\vec{X}_A \oplus \vec{T}_A \oplus \vec{o}_T)$.
7. Verification of secrecy: Alice compares Q_X^{obs} with the predefined value Q_X. If $Q_X^{\mathrm{obs}} + \gamma(Q_X^{\mathrm{obs}}) > Q_X + \gamma(Q_X)$ Alice sets $v_A = 1$, otherwise she sets $v_A = 0$.

8. ACKAEC (Protocol 9): Alice broadcasts $L(1-p)h(Q_Z)$ bits of error-correction (EC) information, in order for the Bobs to correct their raw keys and match Alice's. Additionally, she broadcasts a hash of $\log_2 \frac{n-1}{\varepsilon_{\mathrm{EC}}}$ bits so that each Bob can verify the success of the EC procedure. Alice's broadcasts are encrypted and only the Bobs can decrypt them. If the EC or the verification of secrecy failed, the participants abort the protocol, but this information is encrypted and only available to them.
9. PA: the public randomness outputs a two-universal hash function that maps keys of length $L(1-p)$ to keys of length ℓ, where ℓ is given by

$$\ell = L(1-p)\left[1 - h\left(Q_X + \gamma(Q_X)\right)\right] - 2\log_2 \frac{1}{2\varepsilon_{\mathrm{PA}}}. \tag{D1}$$

Alice and each Bob$_l$ apply the two-universal hash function on their error-corrected keys and obtain the secret conference keys $\vec{k}_A$ and $\vec{k}_{B_l}$. However, if the protocol aborted in the previous step, they do nothing.

We remark that, in order to fairly compare the performance of Protocol 1, for the plots we consider the net number of generated bits, i.e., the number of conference key bits produced by one execution of the protocol, minus the bits consumed from previously established conference keys. The latter amounts to $|\vec{k}_1| + |\vec{k}_2| + |\vec{k}_3| + n$ bits, which yield a net conference key length given by Eq. (1).

Since one of the major novelties of our ACKA protocol is the efficient parameter estimation (or source verification) in step 6 of Protocol 1, we would like to spend a few words on its functioning.

First of all, we emphasize that the parameter Q_X^{obs} estimates the phase error rate of the state distributed by the untrusted source. If in a test round a GHZ state is distributed, the Parity of the outcomes obtained by measuring every qubit in the X basis is zero: $\oplus_{t=1}^n X_t = 0$. Thus these instances do not contribute to the error rate Q_X^{obs}. As a matter of fact, one can simplify the error rate expression provided in step 6 by noting that $\vec{o}_T = \vec{T}_A \oplus (\oplus_{t\neq A}\vec{X}_t)$, which substituted in the error rate yields $Q_X^{\mathrm{obs}} = \omega_r(\oplus_{t=1}^n \vec{X}_t)$.

The reason for which we require Alice to input a random bit T_A in place of her outcome X_A in the Parity protocol is that, in this way, we prevent any dishonest party from artificially decreasing the error rate ($Q_X^{\mathrm{obs}} = \omega_r(\oplus_{t=1}^n \vec{X}_t)$) based on the X_t outcomes of all the other parties. Indeed, if Alice would input X_A, a dishonest party who is the last to broadcast in the Parity protocol can arbitrarily set the output of Parity—and thus $\oplus_{t=1}^n X_t$—to zero.

Finally, we employ the Parity protocol to compute the Parity $\vec{o}_T = \vec{T}_A \oplus (\oplus_{t\neq A}\vec{X}_t)$ instead of using a regular broadcast in order to preserve the participants' anonymity.

To see why their identities would be under threat, let us suppose that we replace the Parity protocol with a regular broadcast. Then, based on the state that Eve—who controls the source—distributed in a given test round, she could make predictions on the X outcomes of the parties and compare them with the broadcast bits. Since Alice always broadcasts a random bit instead of her X outcome as the other parties do, Eve could distinguish her broadcast and learn her identity. Similarly, if a Bob did not measure in the X basis for some test round in $\vec{k}'_T$ (due to a mismatch between $\vec{k}'_T$ and his testing key), he must broadcast a random bit and Eve could learn his identity.

Protocol 2: Fully anonymous conference key agreement (**fully ACKA**)

TABLE I. Parameters used in Protocols 1 and 2.

L	total number of GHZ states distributed and detected (including noisy states)
p	probability of a test round (typically $p \leq 0.01$)
Q_Z	estimation of the largest error rate between the Z outcomes of Alice and of any Bob
Q_X	threshold value of the test error rate, picked in the interval $[0, 1/2)$ and defined as the frequency of nonpassed test rounds. A test round is passed if $\oplus_{t=1}^n X_t = 0$, where $X_t \in \{0,1\}$ is party t's outcome in the X basis mapped to a binary value
v_A, v_t, and v_{B_l}	verification bits of Alice, a generic party $t \in \{1,\ldots,n\}$ and Bob$_l$ (for $l = 1,\ldots,m$), respectively
r_V	number of iterations in the Veto protocol (Protocol 4)
r_N	number of iterations in the TKD protocol (Protocol 8)
$\varepsilon_{\mathrm{EC}}$	failure probability of the error-correction subprotocol
$\varepsilon_{\mathrm{enc}}$	failure probability of the AMD code (F, G) [20,39]
$\gamma(Q_X)$	statistical fluctuation, defined as the positive root of the equation [10,40]: $\ln\binom{L(1-p)\gamma + LQ_X}{LpQ_X} + \ln\binom{L(1-Q_X)-L(1-p)\gamma}{Lp(1-Q_X)} = \ln\binom{L}{Lp} + 2\ln\varepsilon_x$
ε_x^2	upper bound on the probability that the test error rate affecting the key-generation rounds is larger than the observed test error rate (Q_X^{obs}) corrected by the statistical fluctuation γ
$\varepsilon_{\mathrm{PA}}$	probability related to the success of privacy amplification

1. The parties perform the fully ACKAID protocol (Protocol 7). If the fully ACKAID protocol does not abort, Alice is guaranteed to be the only sender and the Bobs are notified to be receivers, except for a small probability.
2. Alice generates a random bitstring of length L where 1 corresponds to a test round and 0 to a key generation round. Given that p is the probability that a round is identified as a test round, she compresses the string to a testing key $\vec{k}_T$ of length $Lh(p)$. Additionally, Alice generates the strings $\vec{r}_l = F(b_l, \vec{r}_\emptyset)$ (for $l \in \{1, \ldots, m\}$) where (F, G) is an AMD code and $(b_l, \vec{r}_\emptyset)$ is a concatenated random string.
3. The parties perform the TKD protocol (Protocol 8) in order to distribute the key $\vec{k}_l = (\vec{k}_T, \vec{r}_l)$, which includes the testing key $\vec{k}_T$, to every Bob.
4. Repeat the following for L rounds.

   4.1. An n-party GHZ state is distributed to the n parties.

   4.2. Alice and the Bobs measure their qubits according to the testing key $\vec{k}_T$. They measure in the Z basis if the round is a key generation round, or in the X basis if the round is a test round. All the other parties measure X.

5. Once all the qubits have been measured (bounded storage assumption), the testing key $\vec{k}_T$ is publicly revealed. This is done anonymously by iterating the Parity protocol (Protocol 3) $Lh(p)$ times. In each instance of the Parity protocol, Alice inputs a bit of $\vec{k}_T$, while the other parties input 0. The output of the Parity protocols is $\vec{k}'_T$.
6. For every round in step 4 that is labeled as a test round by $\vec{k}'_T$, the n parties perform the Parity protocol with the following inputs. Let X_t (for $t \in \{1, \ldots, n\}$) be the outcome of party t if they measured X in that round, otherwise $X_t \in_R \{0, 1\}$. Every party except for Alice ($t \neq A$) inputs X_t, while Alice inputs $T_A \in_R \{0, 1\}$. Let $\vec{o}_T$ be the output of the Parity protocols for all the test rounds in $\vec{k}'_T$. Alice computes $Q_X^{\text{obs}} = \omega_r(\vec{X}_A \oplus \vec{T}_A \oplus \vec{o}_T)$.
7. Verification of secrecy: Alice compares Q_X^{obs} with the predefined value Q_X and sets $v_s = 1$ if $Q_X^{\text{obs}} + \gamma(Q_X^{\text{obs}}) > Q_X + \gamma(Q_X)$ and $v_s = 0$ otherwise. All the parties perform Veto (Protocol 4), where Alice inputs $v_s \vee v_A$, Bob_l inputs v_{B_l} and the nonparticipants input 1 if they have not been notified in TKD, otherwise they input 0. If Veto outputs 1, the protocol aborts for every party.
8. Fully ACKAEC (Protocol 10): Alice anonymously broadcasts $L(1-p)h(Q_Z)$ bits of EC information, in order for the Bobs to correct their raw keys and match Alice's. Additionally, she anonymously broadcasts a hash of $\log_2 \frac{n-1}{\varepsilon_{\text{EC}}}$ bits so that each Bob can verify the success of the EC procedure. If the EC fails for at least one Bob, the protocol aborts but this information is only available to Alice and the Bobs.
9. PA: the public randomness outputs a two-universal hash function that maps keys of length $L(1-p)$ to keys of length ℓ, where ℓ is given by

$$\ell = L(1-p)\left[1 - h\left(Q_X + \gamma(Q_X)\right) - h(Q_Z)\right] - \log_2 \frac{2(n-1)}{\varepsilon_{\text{EC}}} - 2\log_2 \frac{1}{2\varepsilon_{\text{PA}}}. \tag{D2}$$

   Alice and each Bob_l apply the two-universal hash function on their error-corrected keys and obtain the secret conference keys $\vec{k}_A$ and $\vec{k}_{B_l}$. However, if the protocol aborted in the previous step, they do nothing.

APPENDIX E: PROTOCOLS WITHOUT MULTIPARTITE ENTANGLEMENT

In order to evaluate the benefits of using GHZ states to perform ACKA and fully ACKA, we develop alternative protocols, which rely only on bipartite private channels, implemented with Bell pairs. For this reason, we denote these protocols as bipartite ACKA and bifully ACKA, respectively. We remark that bipartite ACKA and bifully ACKA are not mere extensions of the Anonymous Message Transmission protocol [20] to multiple parties, as they are thoroughly optimized to achieve the tasks under consideration.

Protocol 11: ACKA without multiparty entanglement (**bipartite ACKA**)

1. The parties perform the ACKAID protocol (Protocol 6). If the ACKAID protocol does not abort, Alice is guaranteed to be the only sender and the Bobs learn the identities of Alice and of each other, except for a small probability.
2. Alice generates uniformly at random an L_b-bit conference key, $\vec{k}_A$.
3. Every party—except for Alice—sends a random string of L_b bits to every other party through the bipartite private channels. Alice sends the conference key $\vec{k}_A$ to the Bobs and a random string to the other parties. Bob_l identifies the string $\vec{k}_A$ received from Alice as his conference key: $\vec{k}_{B_l} = \vec{k}_A$, for $l \in \{1, \ldots, m\}$.

Protocol 12: Fully ACKA without multiparty entanglement (**bifully ACKA**)

1. The parties perform the fully ACKA$_{\text{ID}}$ protocol (Protocol 7). If the fully ACKA$_{\text{ID}}$ protocol does not abort, Alice is guaranteed to be the only sender and the Bobs are notified to be receivers, except for a small probability.
2. Alice generates uniformly at random an L_b-bit conference key, $\vec{k}_A$, and encodes it as $F(\vec{k}_A)$ with an AMD code (F, G), such that $|F(\vec{k}_A)| = L_b + 2(\log_2 L_b + \log_2 1/\varepsilon_{\text{enc}})$. She also sets her verification bit to $v_A = 0$.
3. Repeat for $n-1$ times.
   - 3.1. Alice randomly picks a party $s \in \{1, \ldots, n\}$ that has not been notified in previous iterations.
   - 3.2. The n parties perform the Notification protocol (step 2 in Protocol 8), such that party s is notified with high probability.
   - 3.3. The n parties execute $|F(\vec{k}_A)|$ rounds of the Parity protocol with the following inputs. If the notified party s corresponds to Bob$_l$ (for $l \in \{1, \ldots, m\}$), Alice inputs $F(\vec{k}_A)$, Bob$_l$ inputs a random bitstring $\vec{r}$, and the other parties input $\vec{0}$. Otherwise, if s is a nonparticipant, Alice inputs $F(\vec{k}_A) \oplus \vec{r}$, where $\vec{r}$ is a random string, and the other parties input $\vec{0}$. Let $\vec{o}$ be the output of the Parity protocols.
   - 3.4. If s corresponds to Bob$_l$, he retrieves the conference key by computing $\vec{k}_{B_l} = G(\vec{r} \oplus \vec{o})$. If $\vec{k}_{B_l} = \top$, then he sets $v_{B_l} = 1$, otherwise $v_{B_l} = 0$. If s is a nonparticipant, Alice computes $G(\vec{o} \oplus \vec{r})$. If the computation returns $\top$, then Alice sets $v_A = 1$.
4. Bob$_l$ (for $l \in \{1, \ldots, m\}$) sets $v_{B_l} = 1$ if he has not been notified in step 3. The n parties perform Veto (Protocol 4) where Bob$_l$ inputs v_{B_l} (for $l \in \{1, \ldots, m\}$), Alice inputs v_A and the other parties input 1 if they have not been notified in step 3, otherwise they input 0. If Veto outputs 1, the protocol aborts for every party and the participants set $\vec{k}_A = \vec{k}_{B_l} = \emptyset \ \forall l$.

We emphasize that the bifully ACKA protocol (Protocol 12) satisfies the anonymity condition for fully ACKA protocols (B14) given in Definition 6, opposed to the fully ACKA protocol based on GHZ states (Protocol 2), which satisfies a weaker anonymity property (Definition 7). More specifically, the bifully ACKA protocol is ε_{an}-anonymous with respect to (dishonest) receivers in the sense of conditions (B9) and (B10), i.e., the reduced state of the receivers is close to a state independent of the identity of the other participants. Conversely, Protocol 2 satisfies a weaker anonymity condition with respect to the receivers, formalized in terms of their guessing probability by conditions (B16) and (B17).

Moreover, note that the security of bipartite ACKA and bifully ACKA can be proved without the assumption about the n parties holding a short-lived quantum memory (bounded storage assumption), while this assumption is crucial for the security of the ACKA and fully ACKA protocols (Protocols 1 and 2), as discussed in Sec. III.

Theorem 9 (Security of bipartite ACKA and bifully ACKA): The bipartite ACKA protocol (Protocol 11), exclusively based on bipartite private channels, yields a secret conference key of length L_b and is ε_{tot}-secure according to Definition 6, with $\varepsilon_{\text{tot}} = 2^{-r_V} + (n-1)\varepsilon_{\text{enc}}$.

The bifully ACKA protocol (Protocol 12), exclusively based on bipartite private channels, yields a secret conference key of length L_b and is ε_{tot}-secure according to Definition 6, with $\varepsilon_{\text{tot}} = 3 \times 2^{-r_V} + (n-1)(2^{-(r_N-1)} + 3\varepsilon_{\text{enc}})$.

[1] J. P. Dowling and G. J. Milburn, Quantum technology: the second quantum revolution, Philos. Trans. R. Soc. A **361**, 1655 (2003).

[2] I. H. Deutsch, Harnessing the Power of the Second Quantum Revolution, PRX Quantum **1**, 020101 (2020).

[3] European Commission, The quantum flagship. https://qt.eu [Online].

[4] C. H. Bennett and G. Brassard, in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing* (IEEE, Bangalore, India, 1984), p. 175.

[5] E. Diamanti, H.-K. Lo, B. Qi, and Z. Yuan, Practical challenges in quantum key distribution, npj Quantum Inf. **2**, 16025 (2016).

[6] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. S. Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden, Advances in quantum cryptography, Adv. Opt. Photonics **12**, 1012 (2020).

[7] I. Q. Technology, Quantum key distribution (QKD) markets: 2019–2028. https://www.insidequantumtechnology.com/product/quantum-key-distribution-qkd-markets-2019-2028 [Online].

[8] M. Epping, H. Kampermann, C. Macchiavello, and D. Bruß, Multi-partite entanglement can speed up quantum key distribution in networks, New J. Phys. **19**, 093012 (2017).

[9] F. Grasselli, H. Kampermann, and D. Bruß, Finite-key effects in multipartite quantum key distribution protocols, New J. Phys. **20**, 113014 (2018).

[10] F. Grasselli, H. Kampermann, and D. Bruß, Conference key agreement with single-photon interference, New J. Phys. **21**, 123002 (2019).

[11] Y. Wu, J. Zhou, X. Gong, Y. Guo, Z.-M. Zhang, and G. He, Continuous-variable measurement-device-independent multipartite quantum communication, Phys. Rev. A **93**, 022325 (2016).
[12] Z. Zhang, R. Shi, and Y. Guo, Multipartite continuous variable quantum conferencing network with entanglement in the middle, Appl. Sci. **8**, 1312 (2018).
[13] R. L. C. Ottaviani, C. Lupo, and S. Pirandola, Modular network for high-rate quantum conferencing, Commun. Phys. **2**, 118 (2019).
[14] G. Murta, F. Grasselli, H. Kampermann, and D. Bruß, Quantum conference key agreement: A review, Adv. Quantum Technol. **3**, 2000025 (2020).
[15] X.-Y. Cao, J. Gu, Y.-S. Lu, H.-L. Yin, and Z.-B. Chen, Coherent one-way quantum conference key agreement based on twin field, New J. Phys. **23**, 043002 (2021).
[16] Z. Li, X.-Y. Cao, C.-L. Li, C.-X. Weng, J. Gu, H.-L. Yin, and Z.-B. Chen, Finite-key analysis for quantum conference key agreement with asymmetric channels, Quantum Sci. Technol. **6**, 045019 (2021).
[17] S. Das, S. Bäuml, M. Winczewski, and K. Horodecki, Universal Limitations on Quantum Key Distribution over a Network, Phys. Rev. X **11**, 041016 (2021).
[18] M. Proietti, J. Ho, F. Grasselli, P. Barrow, M. Malik, and A. Fedrizzi, Experimental quantum conference key agreement, Sci. Adv. **7**, eabe0395 (2021).
[19] F. Grasselli, *Quantum Cryptography* (Springer International Publishing, Basel, 2021).
[20] A. Broadbent and A. Tapp, in *Advances in Cryptology—ASIACRYPT 2007*, edited by K. Kurosawa (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007), p. 410.
[21] N. Walk and J. Eisert, Sharing Classical Secrets with Continuous-Variable Entanglement: Composable Security and Network Coding Advantage, PRX Quantum **2**, 040339 (2021).
[22] D. Lago-Rivera, S. Grandi, J. V. Rakonjac, A. Seri, and H. de Riedmatten, Telecom-heralded entanglement between multimode solid-state quantum memories, Nature **594**, 37 (2021).
[23] M. Pompili, S. L. N. Hermans, S. Baier, H. K. C. Beukers, P. C. Humphreys, R. N. Schouten, R. F. L. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse, S. Wehner, and R. Hanson, Realization of a multinode quantum network of remote solid-state qubits, Science **372**, 259 (2021).
[24] S. K. Joshi, D. Aktas, S. Wengerowsky, M. Lončarić, S. P. Neumann, B. Liu, T. Scheidl, G. C. Lorenzo, Željko Samec, L. Kling, A. Qiu, M. Razavi, M. Stipčević, J. G. Rarity, and R. Ursin, A trusted-node-free eight-user metropolitan quantum communication network, Sci. Adv. **6**, eaba0959 (2020).
[25] Y.-A. Chen, *et al.*, An integrated space-to-ground quantum communication network over 4600 km, Nature **589**, 214 (2021).
[26] M. Christandl and S. Wehner, in *Advances in Cryptology—ASIACRYPT 2005*, edited by B. Roy (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005), p. 217.
[27] V. Lipinska, G. Murta, and S. Wehner, Anonymous transmission in a noisy quantum network using the *W* state, Phys. Rev. A **98**, 052320 (2018).
[28] G. Brassard, A. Broadbent, J. Fitzsimons, S. Gambs, and A. Tapp, in *Advances in Cryptology—ASIACRYPT 2007*, edited by K. Kurosawa (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007), p. 460.
[29] A. Unnikrishnan, I. J. MacFarlane, R. Yi, E. Diamanti, D. Markham, and I. Kerenidis, Anonymity for Practical Quantum Networks, Phys. Rev. Lett. **122**, 240501 (2019).
[30] Y.-G. Yang, Y.-L. Yang, X.-L. Lv, Y.-H. Zhou, and W.-M. Shi, Examining the correctness of anonymity for practical quantum networks, Phys. Rev. A **101**, 062311 (2020).
[31] F. Hahn, J. de Jong, and A. Pappa, Anonymous Quantum Conference Key Agreement, PRX Quantum **1**, 020325 (2020).
[32] C. Thalacker, F. Hahn, J. de Jong, A. Pappa, and S. Barz, Anonymous and secret communication in quantum networks, New J. Phys. **23**, 083026 (2021).
[33] A. Pappa, A. Chailloux, S. Wehner, E. Diamanti, and I. Kerenidis, Multipartite Entanglement Verification Resistant against Dishonest Parties, Phys. Rev. Lett. **108**, 260502 (2012).
[34] M. Movahedi, J. Saia, and M. Zamani, Secure anonymous broadcast, ArXiv:1405.5326 (2014).
[35] C. Portmann and R. Renner, Cryptographic security of quantum key distribution, ArXiv:quant-ph/1409.3525 (2014).
[36] C. Portmann and R. Renner, Security in quantum cryptography, ArXiv:2102.00021 (2021).
[37] J. Bouda and J. Sprojcar, in *2007 First International Conference on Quantum, Nano, and Micro Technologies (ICQNM'07)* (IEEE, Guadeloupe, French Caribbean, 2007), p. 12.
[38] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PRXQuantum.3.040306 for security proofs and further finite-key analysis.
[39] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs, in *Advances in Cryptology—EUROCRYPT 2008*, edited by N. Smart (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), p. 471.
[40] H.-L. Yin and Z.-B. Chen, Finite-key analysis for twin-field quantum key distribution with composable security, Sci. Rep. **9**, 17113 (2019).
[41] K. Wright, *et al.*, Benchmarking an 11-qubit quantum computer, Nat. Commun. **10**, 5464 (2019).
[42] I. Damgard, S. Fehr, L. Salvail, and C. Schaffner, in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)* (IEEE, Pittsburgh, Pennsylvania, 2005), p. 449.
[43] A. Broadbent and C. Schaffner, Quantum cryptography beyond quantum key distribution, Des. Codes Cryptogr. **78**, 351 (2016).
[44] W. Dür, G. Vidal, and J. I. Cirac, Three qubits can be entangled in two inequivalent ways, Phys. Rev. A **62**, 062314 (2000).
[45] R. Renner, Security of quantum key distribution, Int. J. Quantum Inf. **06**, 1 (2008).
[46] G. Brassard and L. Salvail, in *Advances in Cryptology—EUROCRYPT '93*, edited by T. Helleseth (Springer

Berlin Heidelberg, Berlin, Heidelberg, 1994), p. 410.

[47] M. Tomamichel and A. Leverrier, A largely self-contained and complete security proof for quantum key distribution, Quantum **1**, 14 (2017).

[48] M. Tomamichel, C. Schaffner, A. Smith, and R. Renner, Leftover hashing against quantum side information, IEEE Trans. Inf. Theory **57**, 5524 (2011).

[49] M. Tomamichel and R. Renner, Uncertainty Relation for Smooth Entropies, Phys. Rev. Lett. **106**, 110506 (2011).

[50] M. Tomamichel, C. C. W. Lim, N. Gisin, and R. Renner, Tight finite-key analysis for quantum cryptography, Nat. Commun. **3**, 634 (2012).

[51] M. Tomamichel, *Quantum Information Processing with Finite Resources* (Springer International Publishing, Basel, 2016).