The Power of Random Choice: Optimising Quorum Selection in Anonymous Committees

THEA GRANSTRÖM & CONRAD OLSSON

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

The Power of Random Choice: Optimising Quorum Selection in Anonymous Committees

A Hypergeometric Approach to Quorum Selection in Distributed Consistency Protocols

THEA GRANSTRÖM & CONRAD OLSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

The Power of Random Choice: Optimising Quorum Selection in Anonymous Committees
A Hypergeometric Approach to Quorum Selection in Distributed Consistency Protocols
THEA GRANSTRÖM & CONRAD OLSSON

Supervisor: Elena Pagnin, Computer Science and Engineering
Examiner: Rhouma Rhouma, Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

The Power of Random Choice: Optimising Quorum Selection in Anonymous Committees
A Hypergeometric Approach to Quorum Selection in Distributed Consistency Protocols
THEA GRANSTRÖM & CONRAD OLSSON
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Distributed systems that rely on consistency guarantees, such as Key Transparency and Certificate Transparency, face tension between security and efficiency when using anonymous committees. In such protocols, there are two key parameters: the selection probability p, which determines expected committee size, and the quorum threshold Q, the minimum number of endorsements required to accept a state. The Tail-Hammer framework already improves conservative Chernoff-based bounds by using tighter binomial tail estimates. This thesis refines the analysis further by observing that quorum selection from a randomly formed committee can be modelled almost identically to selecting directly from the full population, a consequence of the hypergeometric distribution collapsing to a simple ratio of binomial coefficients when the quorum is drawn without replacement from the committee. This closed-form simplification yields significantly tighter probability bounds with no loss in security. For the *at least one honest* reliable broadcast setting with 128-bit security and a population of $N = 10^9$, the approach reduces the quorum size from $Q = 481$ to $Q = 74$ and the expected committee size from $C = 828$ to $C = 249$, reductions of approximately 85% and 70% respectively.

The analysis is extended to both reliable broadcast and asynchronous network settings, and to both *at least one honest* and *honest majority* quorum requirements. To handle a potentially malicious central server capable of selectively withholding votes, a new parameter S is introduced, representing the minimum number of votes a participant must receive before accepting a state. Security and liveness conditions on S are derived and analysed, revealing that the parameters p and Q optimised independently are not always jointly compatible with a feasible S, motivating a joint optimisation as future work. All results are implemented in Python using the `gmpy2` library for arbitrary-precision arithmetic. To independently validate the numerical results, the *at least one honest* reliable broadcast setting is re-implemented in R using the `Rmpfr` package, with both implementations producing identical parameters across all tested population sizes.

Keywords: anonymous committees, quorum selection, hypergeometric distribution, distributed systems, key transparency, Byzantine fault tolerance, verifiable random functions, probabilistic security analysis

Acknowledgements

We would first and foremost like to thank our supervisor, Elena Pagnin, for helping us develop the idea for this thesis. We also want to extend a huge thank you to Elena Pagnin and Lucia Lavagnino for their support throughout this project, including many interesting discussions and invaluable guidance.

We would also like to thank Kasper Karlsson and Björn Larsson at Omegapoint for providing us with the resources needed to complete this project and for their continuous support throughout the process.

Finally, we are deeply grateful to our family and friends for their endless encouragement and support during this project. Their patience and motivation have meant a great deal to us throughout this journey.

Thea Granström and Conrad Olsson, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CoD	Consistency-or-Die
CM	Channel Maintainer
CT	Certificate Transparency
KES	Key Evolving Signatures
VRF	Verifiable Random Function
VKD	Verifiable Key Directories
KT	Key Transparency
PMF	Probability mass function
CDF	Cumulative distribution function

Nomenclature

Below is the nomenclature of variables that have been used throughout this thesis.

Variables

N	Number of total participants in the system
f_M	Fraction of malicious participants in the system
M	Number of malicious participants in the system
H	Number of honest participants in the system
I	Number of inactive participants in the system
p	Selection probability in Verifiable Random Function
Q	Quorum size
B_N	Binomial approximation of the committee
B_M	Binomial approximation of malicious participants in the committee
B_H	Binomial approximation of honest participants in the committee
B_I	Binomial approximation of inactive participants in the committee

Contents

List of Figures

List of Tables

1
Introduction

The security in many distributed systems today depends on participants observing the same state of the system's components at the same time. These systems may log public keys, certificates, or transaction histories. Examples of such systems include Certificate Transparency (CT), which provides publicly verifiable logs of web certificates, Key Transparency (KT), which enables users of messaging applications such as WhatsApp or iMessage to verify public keys, and blockchain consensus protocols such as Algorand, which maintain a consistent transaction history across distributed participants [22, 27, 16, 28, 3]. If these systems fail, participants may observe different states. A possible consequence can be that malicious parties present false information to participants without being detected. One notable example of this is in 2011 when fraudulent certificates for Google, Yahoo!, Skype, and others were successfully issued through a compromise of the Comodo Group, enabling man-in-the-middle attacks [23].

In these kinds of systems, there is usually a central server that manages and distributes information to all participants. One alternative is to trust this server to verify the information it holds. However, this makes the server a natural target for attackers, and requires all participants to trust a single authority. An alternative is to use anonymous committees, where a random subset of participants is selected to collectively agree on one correct state that the server provides. Because the selection is random and unpredictable, it becomes more difficult for an adversary to target or corrupt the selected members. This approach is used in blockchain consensus protocols such as Algorand, Ouroboros, and GearBox, and in KT protocols such as Consistency-or-Die (CoD) [16, 20, 11, 8]. A real-world deployment of a cryptocurrency protocol using multiple anonymous random committees is Algorand [1], which is analysed by Blum et al. [5].

For protocols that use anonymous committees, two key parameters must be set: the probability p of being selected into the committee, and the quorum threshold Q, which is the number of committee members that must agree for a state to be accepted as valid. If Q is too small, an adversary has a greater chance of controlling enough committee members to present a false state. If Q is too large, the probability that the committee size falls below the quorum threshold increases, which can cause the protocol to stall. Larger values of Q also increase the computational cost, as participants must verify that committee selection and verification are performed correctly. Choosing p and Q therefore depends on the probability that a randomly formed committee has a composition that allows an adversary to break the system.

This probability in turn depends on the total number of participants, the fraction that are adversarial, and the value of p.

Based on the analysis of David et al. [10], this thesis aims to refine how the selection probability p and the quorum threshold Q are set for protocols using anonymous committees. Specifically, the following research questions were investigated:

- Can a smaller quorum size Q be used by modelling quorum composition more precisely than with binomial approximations, while having the same security guarantees?
- How can security and liveness guarantees be maintained in the presence of a potentially malicious channel maintainer capable of withholding messages?
- Can the found parameters p and Q be validated independently with an R script?

To answer the first question, the binomial approximation used in prior work to model quorum composition is combined with a hypergeometric model which captures sampling the quorum without replacement from the committee. This gives more accurate probability bounds and smaller Q while achieving the same security level. To address the second question, a parameter S is introduced. This represents the minimum number of votes a participant must receive before accepting a state, thereby capturing the threat of a channel maintainer withholding votes from honest participants.

The analysis is implemented in Python using the gmpy2 library for high-precision numerical computation, and is then re-implemented in R with the Rmpfr package to validate the numerical results through an independent evaluation of the probabilities.

2
Related Work

In this chapter, different systems and protocols using anonymous committees are presented to provide an overview of how they can be applied to achieve consensus. The chapter also introduces You Only Speak Once (YOSO), whose ideas are used in protocols employing anonymous committees. Section 2.2 focuses on a previous analysis aimed at improving the efficiency of anonymous committees while maintaining security. This project builds upon that analysis using a different approach, with the goal of obtaining further insights into the behaviour of anonymous committees.

2.1 Consistency-or-Die

Ensuring both append-only behaviour and global consistency is a central challenge in KT systems. While append-only structures prevent previously stored data from being modified or deleted, consistency requires that all users observe the same state of the log, the verifiable directory linking users to their public keys. Without this guarantee, a malicious service provider could execute a split-view attack by presenting different versions of the log to different users.

Several approaches have been proposed to achieve consistency. Gossip protocols have participants exchange log views directly with one another over out-of-band channels, allowing inconsistencies to be detected through peer comparison, but they do not scale well to large populations of participants [27]. Blockchain-based approaches anchor the log to a public ledger so anyone can verify its state, but every update requires global consensus, which is slow and computationally expensive compared to a centralised log server [16, 20]. External auditing committees delegate verification to a fixed set of independent parties, which avoids both peer-to-peer scaling and consensus overhead, but concentrates trust in a small group of auditors who become natural targets for adversaries [22, 23].

To address these limitations, Brorsson et al. [8] propose Consistency-or-Die (CoD), a protocol that enables users to detect inconsistencies independently and abort the protocol if one is discovered. The core idea is to randomly select a small, and initially undisclosed, committee of users to endorse the current log view. By distributing trust across the entire user base and revealing endorsers only after the view has been delivered, CoD makes targeted corruption significantly harder.

CoD relies on three cryptographic building blocks: Verifiable Random Functions

(VRFs), Verifiable Key Directories (VKDs), and Key Evolving Signatures (KES). VRFs are used to select endorsers by producing pseudorandom outputs together with cryptographic proofs that the output was correctly computed from the input under the secret key, which anyone holding the corresponding public key can verify. A VKD is a verifiably append-only directory of public keys and presents a digest Dig_e of the directory's state at epoch e. KES provides forward security by updating the signing key over time, preventing adversaries from forging signatures for past epochs even if a current key is compromised.

The protocol assumes communication through a central but potentially untrusted Channel Maintainer (CM) and progresses in epochs. Each epoch is divided into four phases:

- Collection (Col): Publishers submit data to the CM.
- Distribution (Distr): The CM constructs a signed view of the log and distributes it to users.
- Certification (Cert): Users evaluate a VRF on a common seed to determine whether they are selected as endorsers. Selected users verify the view, sign it, evolve their signing keys, and send their endorsements to the maintainer.
- Verification (Ver): Users download the collected endorsements and verify that their view is supported by a sufficient quorum. If the quorum requirement is not met, a split-view attack is assumed and the user aborts.

Committee selection is modelled probabilistically by partitioning users into honest, malicious and inactive groups. The protocol's security ultimately depends on the quorum parameter Q, defined as the minimum number of endorsements from the committee required for a user to accept a view as consistent. This value must be chosen carefully to satisfy two conditions. It must be large enough to prevent malicious committee members from forming a quorum on their own to support conflicting views, and it must be small enough that honest users can reliably collect Q endorsements when no attack is taking place. When these conditions hold, CoD ensures that users either share a consistent view of the log or detect the inconsistency and stop participating.

2.2 Tail-Hammer

Protocols that rely on anonymous committees selected using VRFs must account for the fact that committee sizes are probabilistic rather than fixed. Many existing protocols analyse these committees using loose statistical bounds, such as Chernoff bounds, to estimate committee size and honesty guarantees. However, these bounds are typically loose in cryptographic settings and can significantly overestimate the number of committee members required. This, in turn, leads to unnecessary computational and communication overhead.

Tail-Hammer, proposed by David et al. [10], addresses this inefficiency by providing a more precise statistical framework for determining both committee sizes and quorum thresholds in protocols such as CoD. Rather than reasoning about the full committee, Tail-Hammer focuses on identifying a quorum Q, defined as the minimum number of committee responses required to safely proceed. The key insight is that once a quorum of size Q has been reached, additional committee responses can be ignored without weakening security guarantees.

The analysis assumes a VRF-based anonymous committee selection mechanism in which each of the N parties is selected independently with probability p. As a result, the total committee size follows a binomial distribution $\mathrm{Bin}(N, p)$. To reason about security, the population is divided into honest H and malicious M parties. The number of honest and malicious committee members can then be modelled as two independent binomial random variables, one for each subgroup.

Tail-Hammer introduces the split-bell approach, which analyses these two distributions separately. Intuitively, the number of honest and malicious participants selected follows two bell-shaped distributions with different means. A suitable quorum Q is chosen in the region where both of the following events are unlikely: that the total committee is smaller than Q (violating liveness), and that the malicious subgroup alone reaches Q (violating security). This allows the protocol to balance efficiency and security by selecting the smallest quorum that satisfies both constraints with negligible probability of failure.

The exact quorum requirements depend on the network model. In settings with reliable broadcast, smaller quorums are sufficient, as honest parties are guaranteed to receive all messages sent to them. Depending on the protocol's needs, it may be sufficient to ensure that a quorum contains *at least one honest* participant or an *honest majority*. In asynchronous networks, however, message delays can be exploited by an adversary to form multiple disjoint subcommittees, known as shadow subcommittees. Preventing such attacks requires choosing a larger quorum so that malicious parties cannot participate in multiple valid quorums or partition honest users' views.

Tail-Hammer also considers the presence of inactive users, which naturally occurs in large-scale systems due to churn, time zones, or temporary outages. Inactivity can be incorporated into the analysis by reducing the effective number of active parties before computing quorum thresholds. While this approach simplifies the mathematical model, accurately estimating inactivity in practice remains a challenge and directly impacts quorum sizing.

By providing exact tail probability computations and closed-form approximations, Tail-Hammer enables protocols to replace conservative bounds with precise statistical guarantees. This results in smaller quorums, improved efficiency, and clearer security guarantees for protocols that rely on probabilistic anonymous committees. The results from Tail-Hammer are presented in Table 2.1.

Table 2.1: Result table from Tail-Hammer with $f_M = 0.3$ and $C = p \cdot N$ [10]

At least one honest party case

		Reliable Broadcast			Asynchronous Network		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	**828** (833)	**822** (826)	**763** (769)	**4,540** (4,540)	**4,342** (4,349)	**3,117** (3,122)
	Q	**481** (484)	**477** (480)	**441** (445)	**3,688** (3,688)	**3,526** (3,532)	**2,523** (2,528)
60	C	**376** (377)	**374** (376)	**363** (363)	**2,055** (2,050)	**2,017** (2,012)	**1,703** (1,704)
	Q	**218** (219)	**218** (218)	**220** (210)	**1,669** (1,665)	**1,638** (1,634)	**1,381** (1,382)

Honest majority case

		Reliable Broadcast			Asynchronous Network		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	**4,819** (4,814)	**4,589** (4,586)	**3,214** (3,212)	**4,819** (4,810)	**4,589** (4,589)	**3,214** (3,248)
	Q	**3,940** (3,936)	**3,750** (3,748)	**2,614** (2,612)	**3,940** (3,932)	**3,750** (3,750)	**2,614** (2,634)
60	C	**2,180** (2,180)	**2,129** (2,130)	**1,778** (1,780)	**2,180** (2,174)	**2,129** (2,128)	**1,778** (1,784)
	Q	**1,782** (1,782)	**1,740** (1,741)	**1,450** (1,452)	**1,782** (1,776)	**1,740** (1,738)	**1,450** (1,454)

2.3 Algorand

To address the trade-off between latency and transaction confidence in cryptocurrencies, Yossi Gilad et al. [16] propose Algorand, a cryptocurrency designed to confirm transactions within minutes while scaling to millions of users. Algorand achieves this using a Byzantine Agreement protocol, BA^*, which assigns weights to users based on their stake and guarantees consensus as long as more than two-thirds of the total stake is controlled by honest participants.

To scale consensus, Algorand employs cryptographic sortition based on VRFs to privately and randomly select committees of users to execute each step of the protocol. Each user independently determines their selection using a common public seed and their private key, producing a proof that can be verified by others. The number of selected participants follows a binomial distribution determined by the selection probability, and protocol parameters are chosen such that the probability of selecting a sufficiently honest committee is overwhelmingly high.

Committee members participate only once and are replaced at every step, which mitigates adaptive adversaries attempting to target or corrupt participants after their identities are revealed. Since committee membership is not known prior to message propagation, adversaries cannot predict or proactively attack selected users.

Consensus in Algorand is achieved through voting within committees, where decisions depend on whether the number of votes for a value exceeds a predefined fraction of the committee. There are multiple different types of committees to handle different parts of the protocol in the real-world deployment of Algorand [1]. The protocol provides probabilistic guarantees of safety and liveness, relying on carefully chosen committee sizes and thresholds to ensure that adversarial influence remains negligible [5].

2.4 Gearbox

Another relevant protocol is Gearbox, introduced by David et al. [11], which aims to reduce committee sizes in sharded blockchain systems without compromising security. Traditional sharding protocols require large committees to guarantee both safety and liveness under worst-case corruption assumptions, which limits scalability.

Gearbox separates the requirements for safety and liveness. This allows the system to use small committees that preserve safety, while accepting that liveness may temporarily fail. When this happens, for example if the corruption level exceeds the assumed bound, the system detects it through a control chain, a dedicated blockchain component that remains live and safe whose role is to detect liveness failures and trigger corrective actions, and resamples a new committee with updated parameters.

Committee members are selected uniformly at random from the global population, and security depends on the fraction of corrupted participants in the sampled committee. The number of corrupted members is modelled as a random variable arising from sampling, and analysed using probabilistic methods. When sampling without replacement, the number of corrupted members follows a hypergeometric distribution, which allows bounds on the probability that a committee exceeds a given corruption threshold.

By adapting committee sizes based on observed liveness behaviour, Gearbox improves efficiency for the actual corruption level of the network rather than relying only on worst-case assumptions. This shows how committee size and security depend on the distribution of corrupted participants, and how probabilistic analysis can be used to guide parameter choices.

2.5 Blockchains

Benhamouda et al. [4] study how a public blockchain can maintain and use a secret over time using small, dynamic committees. The protocol is based on proactive secret sharing, where a secret is distributed among a committee and re-shared with a new committee in each epoch.

Committee selection is based on verifiable random functions (VRFs), allowing users to privately determine whether they are selected. Members remain anonymous until they act, which limits an adversary's ability to target them in advance. The protocol uses two types of committees: a nominating committee that selects the next committee, and a holding committee that maintains shares of the secret.

A challenge is that small committees are more likely to contain a larger share of corrupted members. To handle this, committee formation is modelled as random sampling from the full set of users. The number of corrupted members in a committee is treated as a random variable, and analysed using binomial distributions

together with standard concentration bounds.

This analysis is used to choose parameters such as committee size and thresholds so that the probability of selecting a committee with too many corrupted members is small. The result shows how security depends on the distribution of corrupted participants in randomly selected committees, and how this affects parameter choices.

2.6 You Only Speak Once

The concept You Only Speak Once (YOSO) was introduced by Gentry et al. to help protocols in distributed systems against strong network adversaries [15]. YOSO does this by randomly assigning roles to machines participating in the protocol. These roles are meant to take care of different contributions to the protocol's work. When the machine is done with the role's work it passes the result over to another party and deletes its own state including any sensitive information. This makes it more difficult for an adversary to target role assigned machines for any useful knowledge. At this point, shutting down the machine with a Denial-of-Service attack to try to destroy the system is useless because the machine was not going to speak again. Also, trying to take over the machine is ineffective as there is nothing left to steal from it.

3
Theory

This chapter introduces the concepts, mathematics and foundation of protocols using anonymous committees to guarantee consistency. It also states the assumptions regarding security and efficiency of these protocols.

3.1 System Model

This thesis considers protocols that use randomly selected anonymous committees to reach agreement on a shared state. Each participant in the system is independently selected into the committee with probability p, and the resulting committee of participants votes on the state provided by a central server, with the goal that all honest participants agree on the same state. Such protocols operate under specific assumptions about the network and the adversary, which determine what guarantees the protocol can provide and directly affect how the quorum parameter Q can be chosen. In practice, systems that rely on anonymous committees may involve participants performing actions beyond voting, and the data they operate on may take forms other than states. The essential requirement is consistency, meaning that all honest participants agree on the same outcome regardless of how the rest of the system is structured. For example, in Key Transparency (KT) the state corresponds to a digest of the key directory [7, 8], while in blockchain protocols it corresponds to a proposed block [16, 20].

3.1.1 Verifiable Random Functions

In this thesis, each participant in the system is assumed to be independently selected into the committee with probability p using a Verifiable Random Function (VRF). This is a pseudorandom function where the owner of a secret key can evaluate the function on an input and produce both a unique output and a cryptographic proof of correctness [29]. Anyone holding the corresponding public key can verify that the output was computed correctly. Observing the output and proof for one input does not compromise the unpredictability of the function at any other input.

VRFs are commonly used in protocols relying on anonymous committees to privately and randomly select committee members [16, 8, 9]. Since the VRF output is pseudorandom, an adversary cannot predict which participants will be selected before they reveal their membership. Committee members remain hidden until they act, which prevents an adversary from targeting or corrupting them in advance. Because

the VRF output is verifiable, any participant can check that a member was selected correctly by verifying the corresponding proof against the public key.

Formally, the most important algorithms for VRF are listed below [12]:

- **KeyGen:** $(sk, pk) \leftarrow \text{KeyGen}(1^\lambda)$
  Input: bit security parameter λ. **Output:** secret key sk, public key pk.
  The secret key is used to generate VRF outputs, while the public key is used by others to verify them.

- **Prove:** $(y, \pi) \leftarrow \text{Prove}(sk, x)$
  Input: secret key sk, input x. **Output:** VRF output y, proof π.
  The proof shows that y was correctly computed from x using the secret key.

- **Ver:** $b \leftarrow \text{Ver}(pk, x, y, \pi)$
  Input: public key pk, input x, output y, proof π. **Output:** $b \in \{0, 1\}$.
  Anyone can verify that the output was correctly generated without knowing the secret key.

These algorithms satisfy three security properties. Correctness requires that if $(y, \pi) \leftarrow \text{Prove}(sk, x)$, then $\text{Ver}(pk, x, y, \pi) = 1$, ensuring that honestly generated proofs are always accepted; since verification requires only the public key pk, anyone can publicly verify the output without knowledge of the secret key. Uniqueness requires that for a fixed public key pk and input x, there exists only one valid output y that can be accepted. Finally, pseudorandomness requires that without knowledge of sk, the output y is computationally indistinguishable from a random value.

3.1.2 Distributed Systems

A distributed system consists of multiple interconnected components working together to produce a single, reliable result. Such systems must be Byzantine fault tolerant, meaning they can continue to operate correctly even when some components malfunction or send conflicting information to different parts of the network [21]. Since faulty or malicious components may behave in unpredictable ways, no single component can be trusted alone. Instead, components must communicate and agree among themselves before accepting any results as correct.

3.1.3 Quorums

A quorum is the minimum number of nodes that must agree for a distributed protocol to accept a decision as valid [25]. The size of the quorum is directly linked to Byzantine fault tolerance [21]. It must be large enough to ensure that an adversary cannot reach the quorum threshold using only malicious participants, ensuring that any valid decision requires the agreement of at least some honest participants.

Setting the quorum size is essentially a trade-off between security and liveness [11]. If the quorum is set too small, security is compromised because an adversary might reach the quorum threshold alone. If the quorum is too large, the system risks

liveness failures. Since real-world systems always have a fraction of nodes being inactive, requiring too many responses means the system may frequently fail to reach a decision.

3.1.4 Network Model

In this thesis, two network models are to be considered: reliable broadcast and asynchronous network. In a reliable broadcast setting, all honest participants receive all messages sent to them [6]. This means that all honest participants observe the same set of votes before deciding whether to accept a state, and the adversary cannot use message delays to manipulate what different participants see. Furthermore, since all participants receive all votes, a malicious committee member cannot cast multiple votes for different states without being detected, as honest participants will observe the duplicate and discard that member's votes. Each committee member therefore casts at most one vote.

In an asynchronous network setting, there are no guarantees on message delivery or ordering [13]. An adversary can delay or drop messages from honest participants, which allows it to partition the committee into separate subsets. These are known as shadow subcommittees, where different participants see different states [10], as illustrated in Figure 3.1. In this setting, the malicious parties also have the power to send several conflicting votes. In the worst case for an honest participant, a quorum of size Q may form entirely from the malicious participants and half of the honest participants, since the adversary can selectively withhold the remaining half. This makes the asynchronous setting strictly harder to defend against, and requires a larger quorum Q to achieve the same security guarantees.

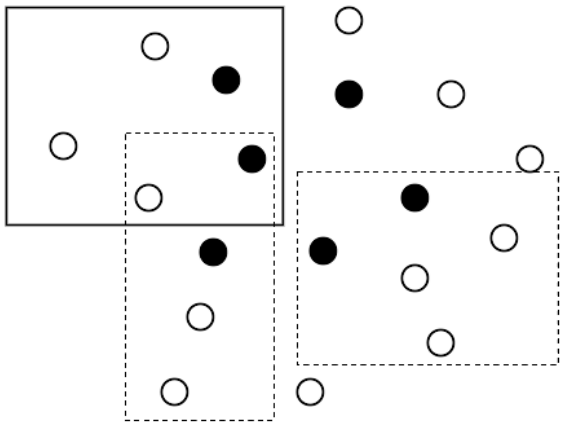

Figure 3.1: Illustration of shadow subcommittees in an asynchronous network. The solid rectangle and the two dashed rectangles each represent a quorum of size Q, containing a mix of honest (unfilled) and malicious (filled) participants. By selectively withholding messages, a malicious Channel Maintainer can present different quorums to different participants, enabling a split-view attack.

To manage time, protocols with anonymous committees use epochs as a unit to process data. In the beginning of each epoch, a number of random participants are selected to become part of the committees and act as voters.

3.1.5 Threat Model

Protocols using anonymous committees typically assume a Byzantine adversary, who controls a subset of M malicious participants out of a total population of N participants, where the fraction of malicious participants is $f_M = M/N$. A Byzantine adversary is not limited to passive eavesdropping. It may cause malicious participants to behave in any way that benefits the adversary, send conflicting messages to different parts of the network, and coordinate all malicious participants to act as a single entity [21, 31].

The adversary is assumed to be static, meaning the set of malicious participants is fixed before each epoch begins [16, 9]. This means the adversary cannot corrupt new participants during a running epoch based on which participants were selected into the committee. This assumption is important for the security of the VRF-based committee selection, since an adaptive adversary could otherwise wait to see who was selected and corrupt them afterwards. This assumption is standard in VRF-based committee selection, and is reasonable in practice since epochs are typically short enough that an adversary cannot identify and corrupt a selected participant before the epoch ends. Tolerating adaptive corruption would require additional mechanisms such as secure erasures or stateless ephemeral roles as in YOSO (Section 2.6), which fall outside the scope of this work.

The remaining $H = N - M$ participants are assumed to be honest, meaning they follow the protocol correctly. However, not all honest participants are assumed to be active in every epoch. Protocols based on anonymous committees often require that a sufficient number of honest participants are available in each epoch, while acknowledging that some honest participants may be temporarily offline or unreachable [20, 16, 8]. An honest participant who does not respond during a given epoch, for example due to being offline or experiencing network disruptions, is considered inactive for that epoch. Such participants neither vote nor participate in verification. The fraction of inactive honest participants directly affects liveness, since a large inactive population reduces the expected number of committee members selected in an epoch and makes it harder to get enough votes to reach the quorum Q [10].

3.2 Transparency and Consistency Systems

This section introduces two systems that rely on consistency and agreement guarantees to function correctly: Key Transparency and Certificate Transparency. Both systems face challenges of making sure all participants observe the same state, making them candidates for protocols that use randomly selected anonymous committees and quorums to reach consistency.

3.2.1 Key Transparency

Key Transparency (KT) is a security model designed to prevent communication service providers from secretly intercepting messages between users in distributed systems [27]. It works by maintaining a verifiable, tamper-evident log of the directory that links usernames to their public keys. Instead of blindly trusting the service provider, key transparency allows the software to automatically monitor accounts to guarantee that the provider is not secretly issuing fake keys or showing different keys to different users.

One of the main problems with key distribution, referred to as trust establishment by Unger et al. [32], is the trade-off between strong security and easy usability (efficiency). Systems that provide the highest level of security, such as requiring users to manually verify long cryptographic fingerprints, demand too much effort from average users and fail to achieve widespread usage. On the other hand, highly convenient systems, like central key directories that work automatically in the background, offer weaker security. These systems, like a central server that hands out keys automatically, are vulnerable to man-in-the-middle attacks if the server is compromised or forced to distribute fake keys. Users blindly trusting these servers as a central authority presents a significant risk since these servers are natural targets for adversaries.

The main goal of Key Transparency (KT) is to provide an automated way for users to verify that they are using the correct public keys, which could remove the need to blindly trust a central service provider [7]. Specifically, KT aims to guarantee these two properties:

- **Append-only**: the key directory can only be added to, existing information can never be modified or deleted. A malicious server or adversary cannot secretly go back and alter, overwrite, or erase the key history.
- **Consistency**: everyone using the service sees the exact same public key for a specific person.

KT aims to minimise the risk of split-view attacks where a malicious server secretly presents distinct, conflicting versions of the key directory to different users. For example, the server might show one user the correct public key for that user's account, but show a fake key to the other users. This allows the server to secretly intercept messages without users ever realising their key was replaced.

3.2.2 Certificate Transparency

To publicly log and audit TLS and SSL certificates, Certificate Transparency (CT) is used to detect mistakenly or maliciously issued certificates that could otherwise go undetected [23]. Before CT, a Certificate Authority (CA) could issue a certificate for a domain to an unauthorised party and nobody, including the owner of the certificate, would necessarily know. CT uses append-only logs making tampering with certificates detectable, monitors watching the logs for suspicious certificates,

and auditors verifying that logs behave honestly. Like KT, CT relies on mechanisms to ensure that all parties observe the same log state, making it a relevant component for the consistency protocols studied in this work.

3.3 Random Sampling

Members of the anonymous committee are selected by randomly sampling from the full population of participants. The statistical properties of this sampling process determine how many malicious participants are likely to end up in the committee, which directly affects the security of the protocol. Two models of random sampling are relevant here. In the VRF-based committee formation used in this thesis, each participant is selected independently with probability p, so the committee size follows a binomial distribution as the sum of N independent Bernoulli trials [18, p. 105–106]. The other model is sampling without replacement from a finite population, where the count of draws from any given subset follows a hypergeometric distribution [33, p. 43–44].

3.3.1 Binomial Distribution

Let $N \in \mathbb{N}$ and $p \in [0, 1]$. Consider a sequence of N independent trials where each trial results in a success with probability p and a failure with probability $1 - p$. Let X be the number of successes across all N trials [18, p. 105–106]. Then

$$X \sim \mathrm{Bin}(N, p),$$

with probability mass function (pmf)

$$\Pr(X = x) = \binom{N}{x} p^x (1-p)^{N-x}, \quad x \in \{0, 1, \ldots, N\}.$$

The cumulative distribution function (cdf) is given by

$$\Pr(X \leq x) = \sum_{j=0}^{x} \binom{N}{j} p^j (1-p)^{N-j}.$$

and the complementary cdf is

$$\Pr(X > x) = \sum_{j=x+1}^{N} \binom{N}{j} p^j (1-p)^{N-j}.$$

The binomial cdf and complementary cdf can then be rewritten into regularised incomplete beta functions I_{1-p} and I_p respectively [30, 8.17(i)]

$$I_{1-p}(N - x, x + 1) = \Pr(X \leq x) = \sum_{j=0}^{x} \binom{N}{j} p^j (1-p)^{N-j},$$

$$I_p(x, N - x + 1) = \Pr(X \geq x) = \sum_{j=x}^{N} \binom{N}{j} p^j (1-p)^{N-j}.$$

Figure 3.2 shows the probability mass functions of two binomial distributions with $N = 20$ trials and with two different probabilities $p_1 = 0.3$ and $p_2 = 0.7$. Figure 3.3 shows the corresponding cumulative distribution functions.

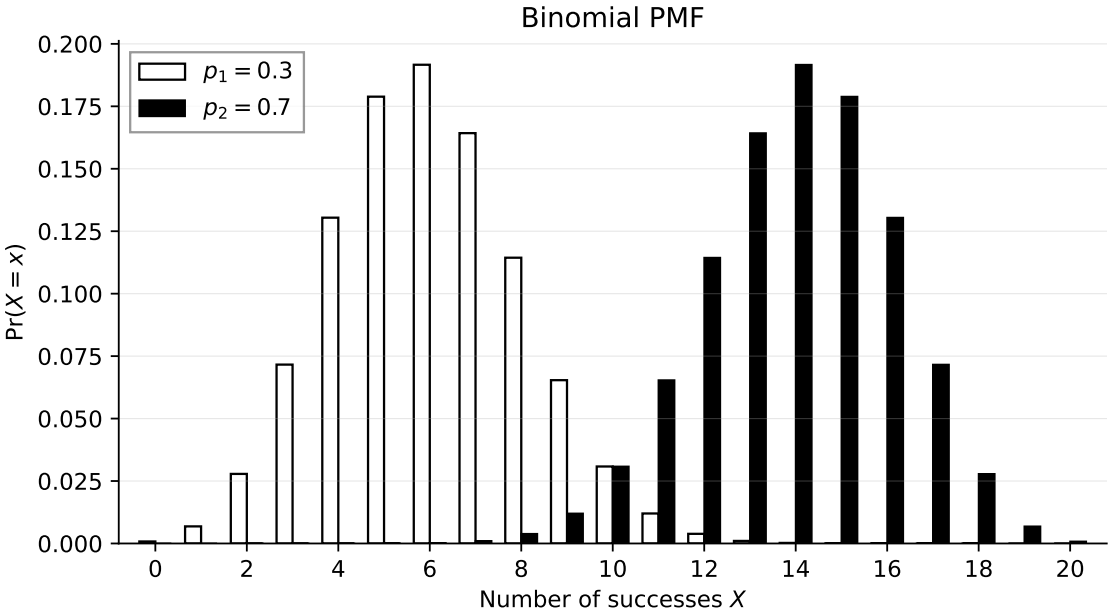

Figure 3.2: Probability mass function of the binomial distribution with $N = 20$ for two different probabilities $p_1 = 0.3$ and $p_2 = 0.7$

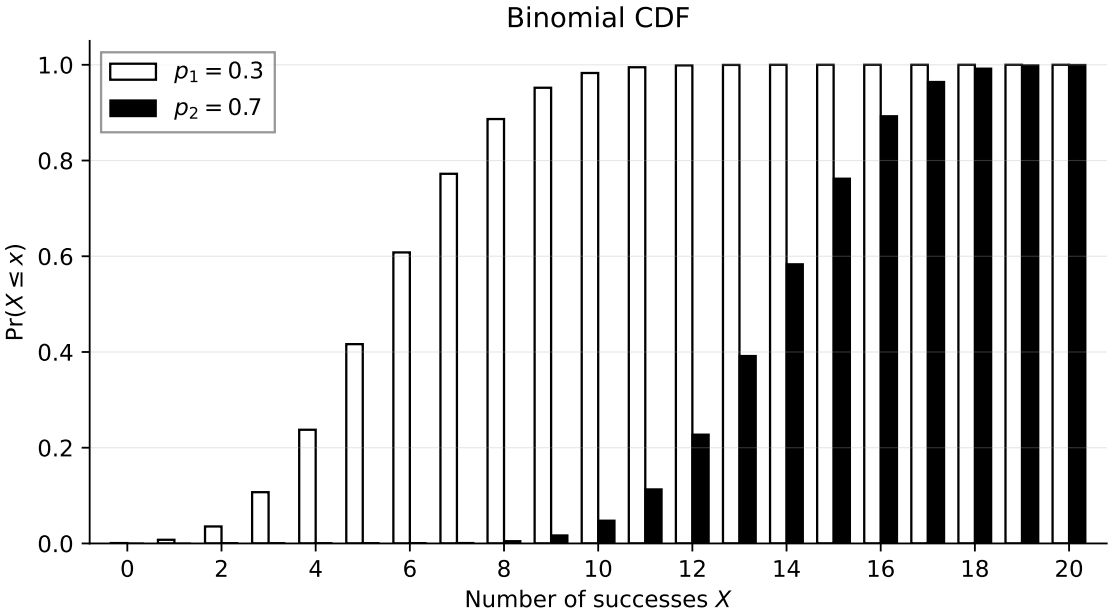

Figure 3.3: Cumulative distribution function of the binomial distribution with $N = 20$ for two different probabilities $p_1 = 0.3$ and $p_2 = 0.7$

To better understand what the binomial distribution means, consider a fair coin-toss where heads occurs with probability $p = 0.5$. If we toss the coin $N = 20$ times and

let X be the number of heads, then $X \sim \text{Bin}(20, 0.5)$. The probability of getting exactly $x = 10$ heads is $\Pr(X = 10) = \binom{20}{10} 0.5^{10} \cdot 0.5^{10} \approx 0.176$, meaning there is roughly an 18% chance of getting exactly 10 heads out of 20 tosses.

3.3.2 Hypergeometric Distribution

Let $N \in \mathbb{N}$ with $N \geq 1$, and let $K, n \in \{0, 1, \ldots, N\}$. Consider a finite population of size N containing exactly K elements labelled as successes and $N - K$ labelled as failures. Let X be a random variable that counts the number of successes in a subset of size n drawn without replacement from the population, where every subset of size n is equally likely. Then X is said to follow a hypergeometric distribution with parameters (N, K, n)

$$X \sim \text{Hyper}(N, K, n),$$

with probability mass function (pmf) [18, p. 237–238]

$$\Pr(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}, \quad max(0, n - N + K) \leq x \leq min(n, K). \tag{3.1}$$

To better understand this, consider an urn containing $N = 20$ balls in total, of which $K = 8$ are red (successes) and $N - K = 12$ are blue (failures). If we draw $n = 5$ balls at random without replacement, and let X be the number of red balls drawn, then $X \sim \text{Hyper}(20, 8, 5)$. The probability of drawing exactly $x = 2$ red balls is $\Pr(X = 2) = \frac{\binom{8}{2}\binom{12}{3}}{\binom{20}{5}} = \frac{28 \cdot 220}{15504} \approx 0.397$, meaning there is roughly a 40% chance of drawing exactly 2 red balls in a sample of 5.

3.3.3 Multivariate Hypergeometric Distribution

Consider a finite population of size N is partitioned into m categories, where category i contains K_i elements with

$$\sum_{i=1}^{m} K_i = N. \tag{3.2}$$

If a sample of size n is drawn uniformly at random without replacement, the probability that the sample contains exactly k_i elements from each category i is [33, p. 47],

$$\prod_{i=1}^{m} \frac{\binom{K_i}{k_i}}{\binom{N}{n}}. \tag{3.3}$$

For example, consider an urn containing $N = 20$ balls in total, partitioned into $m = 3$ colours: $K_1 = 8$ red, $K_2 = 7$ blue, and $K_3 = 5$ green. If we draw $n = 6$ balls at random without replacement, the probability of drawing exactly $k_1 = 3$ red, $k_2 = 2$ blue, and $k_3 = 1$ green is $\frac{\binom{8}{3}\binom{7}{2}\binom{5}{1}}{\binom{20}{6}} = \frac{56 \cdot 21 \cdot 5}{38760} \approx 0.152$, meaning there is roughly a 15% chance of drawing this exact colour composition.

This is an extension of the hypergeometric distribution described in the previous section. Setting $m = 2$ recovers the regular hypergeometric distribution.

3.3.4 Law of Total Probability

Let $\{B_1, \ldots, B_n\}$ be a partition of events of the sample space Ω, meaning the events are mutually exclusive and exhaustive [33, p. 115–117]. Then for any event A,

$$\Pr(A) = \sum_{i=1}^{n} \Pr(A|B_i) \cdot \Pr(B_i) \tag{3.4}$$

the probability of A can be computed by summing the conditional probabilities of A given each partition element, weighted by how likely each case is.

3.4 Libraries

This section presents the libraries used in the implementation and their support for arbitrary-precision arithmetic, which is required for the computations in this work.

3.4.1 Gmpy2

The python library gmpy2 provides access to arbitrary-precision arithmetic by acting as a high-level interface to the underlying C libraries GMP, MPFR, and the MPC [26]. Where MPFR supports multiple-precision floating-point computations with *correct* rounding [14]. The Python library supports the MPFR elementary operations, such as addition, multiplication, and division, which are correctly rounded according to selected rounding rule. While gmpy2 supports high-precision computation on low-level arithmetic primitives, it does not provide a complete set of higher-level special functions as the regularised incomplete beta function and cumulative distribution functions.

3.4.2 Rmpfr

The R package Rmpfr, as gmpy2 in Python, provides arbitrary-precision arithmetic in the R environment based on the MPFR and GMP libraries [24].

Unlike gmpy2, Rmpfr also integrates with R's statistical computing framework. In addition to the basic arithmetic operations, the package provides high-precision implementations of several mathematical and statistical functions, like the regularised incomplete beta function. Since the cumulative distribution function can be expressed in terms of the regularised incomplete beta function, the Rmpfr library can evaluate binomial tail probabilities at arbitrary precision.

3.5 Security Model and Definitions

This section defines the security model used throughout this work, introducing the key concepts and failure events that the protocol must guard against.

3.5.1 Statistical Bit Security Level

A function $\nu : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is said to be *negligible* if for every positive polynomial $\text{poly}(\cdot)$ there exists $b_0 \in \mathbb{N}$ such that for all $b \geq b_0$ [17, p. 16] [19, p. 46],

$$\nu(b) < \frac{1}{\text{poly}(b)}. \tag{3.5}$$

In this work, this definition of a scenario being negligible is applied. Let $b \in \mathbb{N}$ denote the *statistical bit security level* of the protocol. Intuitively, b measures how unlikely bad events are: the larger b is, the smaller the probability 2^{-b} of any undesirable event occurring. All probabilistic guarantees are stated in terms of b, and protocol parameters such as the quorum size Q, and the selection probability p are chosen so that the probability of any security or liveness failure is bounded by 2^{-b}. Since 2^{-b} is negligible in b, this provides the standard cryptographic guarantee that security holds except with negligible probability.

Since the previous analysis of committee selection probabilities and quorum thresholds in Tail-Hammer uses the bit security level of 60 and 128 [10], this analysis uses these as base values for fair comparison.

3.5.2 Security

Security requires that honest participants agree on the state of the system. In Byzantine consensus, this is known as the agreement or safety property [21], and in KT it corresponds to the consistency goal that all participants observe the same key directory [7]. Formally:

A protocol using anonymous committees satisfies security if, for any epoch e whenever two honest participants u_1 and u_2 both accept a state in epoch e, they accept the same state. That is, if s_1 and s_2 denote the states accepted by u_1 and u_2 respectively, then $s_1 = s_2$.

In protocols where anonymous committees provide endorsements, a security violation occurs when the adversary causes at least two conflicting states to each receive enough endorsements to reach the quorum threshold Q.

3.5.3 Liveness

Liveness requires that the protocol continues to make progress, meaning that honest participants are able to accept new states. In distributed systems, a liveness property guarantees that something good eventually happens [2], and in the blockchain setting this is described as the guarantee that honestly submitted data eventually becomes part of the commonly agreed state [20]. Formally:

A protocol using anonymous committees satisfies liveness if the committee formed in each epoch is large enough to meet the quorum requirement Q with overwhelming

probability, ensuring that honest participants can always collect Q valid votes and the protocol continues to make progress [10].

3.5.4 Failure Events

Based on the definitions of security and liveness above, two types of failure events are identified. A security failure occurs when the adversary succeeds in a split-view attack, forming a quorum supporting a conflicting state and causing honest participants to accept different states. A liveness failure occurs when the committee selected in an epoch is too small to reach the quorum threshold, causing the protocol to stall.

The exact form of the security failure event depends on the quorum requirement (Section 3.1.3) and the assumed network model (Section 3.1.4).

A protocol using anonymous committees is said to be b-secure if both failure probabilities are negligible:

$$\Pr[E_{\text{sec}}] \leq 2^{-b} \quad \text{and} \quad \Pr[E_{\text{live}}] \leq 2^{-b} \tag{3.6}$$

3.6 Quorum Security Conditions

In this thesis, an adversary model in which corrupted participants have Byzantine capabilities, meaning they may deviate arbitrarily from the protocol, is considered [21]. The analysis focuses on adversarial strategies that aim to break consistency rather than trying to disrupt liveness, since split-view attacks pose a more severe threat than temporary stalls [10].

In the system, a quorum of size Q is formed from the committee selected during each epoch. The security guarantees depend on the structural properties required from this quorum and on the underlying network assumptions.

Under a *reliable broadcast* setting, all honest participants observe the same set of votes [6]. Under the *at least one honest* requirement, security requires that the probability of an entire quorum consisting only of malicious participants is negligible with respect to the security parameter. Since the protocol requires all Q votes, an adversary can only succeed if every member of the quorum is corrupted. If at least one honest participant is included, that participant will refuse to endorse an incorrect state, preventing the adversary from obtaining all Q votes.

Under the *honest majority* requirement, the quorum must contain strictly more honest than malicious participants. This ensures that adversarial participants cannot dominate the quorum, even if they attempt to deviate from the protocol. In this case, the honest participants collectively determine the outcome according to the protocol rules.

In an *asynchronous network*, the analysis differs from the reliable broadcast case since honest participants may observe different subsets of votes before a quorum is reached [10]. This gives rise to so-called shadow subcommittees, where multiple disjoint subsets of size Q may appear valid. As a result, stronger conditions are required to ensure that malicious participants cannot form conflicting quorums.

3.6.1 Asynchronous network with at least one honest party

We follow the Tail-Hammer analysis for the asynchronous setting [10], where N is the total population, M is the number of malicious participants, and $H = N - M$ is the number of honest participants. The variable p is the probability of being selected into the committee. The relevant conditions from Tail-Hammer are

$$\Pr(B_N < Q) \quad \text{and} \quad \Pr(B_{M+H/2} \geq Q),$$

where $B_N \sim \text{Bin}(N, p)$ represents the committee size, and $B_{M+H/2} \sim \text{Bin}(M + H/2, p)$. In the asynchronous setting, malicious participants can vote for all states simultaneously, while honest participants are assumed to be split evenly between two conflicting states. The second condition therefore ensures that a quorum cannot be formed by the malicious participants together with the honest participants voting for state alone. Both conditions should be negligible: the first one ensures liveness, and the second one ensures that such a shadow subcommittee cannot reach the quorum threshold.

The condition $\Pr(B_{M+H/2} \geq Q)$ is stronger than $\Pr(B_M \geq Q)$, so a separate bound on $\Pr(B_M \geq Q)$ is not needed.

3.6.1.1 Proof.

Write

$$B_{M+H/2} \sim B_M + B_{H/2},$$

where $B_M \sim \text{Bin}(M, p)$ and $B_{H/2} \sim \text{Bin}(H/2, p)$. This follows from the additive property of the binomial distribution. Given that each participant is selected independently with the same probability p, the total number of selected participants from two disjoint groups is the sum of two independent binomial random variables, which is itself binomial [18, p. 50]. Since $B_{H/2}$ is always non-negative, we have

$$B_M \geq Q \implies B_M + B_{H/2} \geq Q.$$

Therefore,

$$\{B_M \geq Q\} \subseteq \{B_{M+H/2} \geq Q\}.$$

By monotonicity of probability, it follows that

$$\Pr(B_M \geq Q) \leq \Pr(B_{M+H/2} \geq Q).$$

Hence, an upper bound on $\Pr(B_{M+H/2} \geq Q)$ automatically controls $\Pr(B_M \geq Q)$, and no separate condition on $\Pr(B_M \geq Q)$ is required.

3.6.2 Optimal Adversarial Strategy in Split-View Attacks

Under the Byzantine adversary model introduced in Section 3.1.5, malicious participants may deviate arbitrarily from the protocol. Specifically, a malicious participant can vote for any number of conflicting states simultaneously, whereas an honest participant only votes for the state they observe. The adversary coordinates all malicious participants as a single entity, choosing their votes and the distribution of honest states to maximise the chance of breaking consistency. The protocol must therefore be analysed against the adversary's optimal strategy, which determines the worst case from the protocol's perspective.

To understand the adversary's optimal strategy with respect to the number of conflicting states, consider a system with $N = 10$ participants, of which $M = 4$ are malicious and $H = 6$ are honest. The malicious participants aim to endorse multiple inconsistent states in order to break consistency, while each honest participant votes for exactly one state.

Suppose the adversary attempts to spread s conflicting states simultaneously. The adversary's best strategy is to distribute the honest states as evenly as possible across the s states, since this maximises the honest votes each state receives. For $s = 2$ states, each state is supported by $H/2 = 3$ honest participants, but for $s = 3$ states, only $H/3 = 2$ honest participants vote for each state (Figure 3.4). Since the adversary benefits from having more honest participants voting for each state, it is preferable to maximise the number of honest participants per state. This means fewer states are strategically better from the adversary's perspective. However, a split-view attack requires at least two conflicting states.

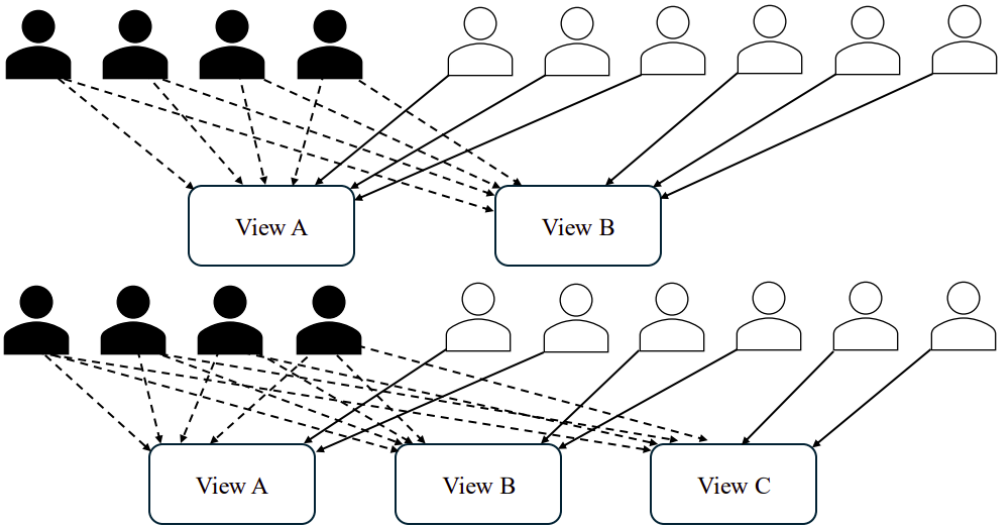

Figure 3.4: The adversary's options with $N = 10, M = 4$ malicious (filled), $H = 6$ honest (unfilled). Malicious participants vote for every state (dashed); each honest participant votes for one (solid). Fewer states give the adversary more honest votes per state, making $s = 2$ optimal.

This trade-off is also reflected in the quorum parameter Q required for security. In the two-state case, a quorum of $Q = 8$ is necessary to ensure that no conflicting state can reach the quorum. In the three-state case, a smaller $Q = 7$ is enough. Consequently, a larger number of states makes the system easier to protect. Each additional state reduces the honest support available per state, lowering the quorum threshold required to detect inconsistency. This is also intuitive: the more states in circulation, the easier it becomes for an honest participant to encounter conflicting votes and identify an ongoing attack.

Consider an adversary propagating $s \geq 2$ conflicting states. By the threat model, each malicious participant can vote for all s states simultaneously, while each honest participant votes for exactly one. The adversary's best strategy distributes the honest votes as evenly as possible, giving each state at most $\lfloor H/s \rfloor$ honest votes. The total votes available per state are therefore at most $M + \lfloor H/s \rfloor$.

Since $\lfloor H/s \rfloor$ decreases as s grows, and the protocol must defend against the adversary's best case, the worst case for the protocol is the smallest value of s. As a split-view attack requires at least two conflicting states, the adversary's optimal strategy is $s = 2$.

4
Methods

This chapter describes the methods used to analyse quorum and committee parameters. First, a probabilistic model for committee composition is developed and expressions for the relevant failure events are derived. These are then used to find optimal values of p and Q, as well as the set size S introduced to handle a potentially malicious channel maintainer. Finally, the computational implementation in Python and R is described.

4.1 Probabilistic quorum analysis

This section analyses quorum selection using probabilistic modelling. The primary objective is to estimate protocol failure probabilities and determine quorum sizes that provide a balance between security and availability. Prior work models committee formation using binomial distributions, where each participant independently gets selected into the committee with probability p, so the number of selected participants in each subgroup is binomially distributed. The security conditions are evaluated directly on the resulting committee composition.

This analysis also models the committee selection with a binomial, but additionally models the selection of a subset of size Q from the formed committee. Since this subset is sampled without replacement from a finite committee, the composition of the sampled subset is therefore behaving as a hypergeometric distribution. This allows security conditions to be evaluated on sampled subsets of the committee, which may in turn permit smaller quorum sizes and committee compositions while maintaining the same security bound.

The methodology consists of formulating probabilistic models for committee composition, deriving expressions for relevant failure events, and optimising parameters based on these probabilities. The analysis is performed under multiple protocol assumptions, including different network models and security conditions.

4.1.1 Assumptions

Table 4.1 shows the probability constraints for security depending on assumed network setting and security condition. All four compositions share the same constraint for liveness: $\Pr(B_N < Q) \leq 2^{-b}$, where $B_N \sim \mathrm{Bin}(N, p)$. Both X and Y are analytical views of the same quorum of size Q drawn uniformly from the same committee,

but they bound different adversarial events. To count the number of malicious participants in the quorum, $X \sim \text{Hyper}(B_N, B_M, Q)$ is used to bound the probability that an adversary controls enough quorum members to break security on their own. Instead, $Y \sim \text{Hyper}(B_N, B_M{+}B_{H/2}, Q)$ treats malicious participants together with half of the honest participants as the effective adversarial population, capturing the shadow subcommittees possible in an asynchronous network (discussed in Section 3.6). Consequently, X alone is sufficient to reason about reliable broadcast, while asynchronous network settings require bounding Y. The derivations of all conditions and the precise definitions of X and Y are given in the following Section 4.1.2.

	Reliable Broadcast	**Asynchronous Network**
At least one honest	$\Pr(X = Q) \leq 2^{-b}$	$\Pr(Y = Q) \leq 2^{-b}$
Honest majority	$\Pr(X \geq \lfloor Q/2 \rfloor) \leq 2^{-b}$	$\Pr(X \geq \lfloor Q/2 \rfloor) \leq 2^{-b}$ and $\Pr(Y = Q) \leq 2^{-b}$

where $X \sim \text{Hyper}(B_N, B_M, Q)$ and $Y \sim \text{Hyper}(B_N, B_M{+}B_{H/2}, Q)$

Table 4.1: The difference in quorum settings between Reliable Broadcast and Asynchronous Network communication models.

4.1.2 Hypergeometric System Model

The system is defined by a total population of N participants. A fraction f_M of these participants is assumed to be malicious, resulting in a total of

$$M = \lfloor f_M \cdot N \rfloor$$

malicious participants, while the remaining $N - M$ participants are honest. In each epoch, a committee is selected uniformly at random from the population, where each participant is selected independently with probability p. Let $B_N \sim \text{Bin}(N, p)$ denote the number of participants in the committee. The quorum is denoted by Q, which represents the number of participants required for the protocol to make progress or reach agreement.

4.1.2.1 Quorum Composition

The number of malicious participants in the quorum is modelled using a hypergeometric distribution. This captures the fact that the quorum is sampled uniformly at random from the committee without replacement, meaning each committee member can appear in the quorum at most once. Prior work [10] models committee composition using independent binomial random variables for the honest and malicious subgroups respectively, and then choosing a quorum size based on the expected malicious and honest in the committee. In this work, given that a committee of size B_N has formed and all votes are assumed to reach each participant, a quorum of size Q is instead drawn uniformly from that committee. The number of malicious members in the quorum is then exactly hypergeometric, conditioned on the committee

composition. Formally:

$$X \sim \text{Hyper}(B_N, B_M, Q),$$

where $B_M \sim \text{Bin}(M, p)$, $B_N \sim \text{Bin}(N, p)$, and $B_N \sim B_M + B_H$.

Since B_N and B_M are random variables (the committee selection is uniformly random), the law of total probability is applied to determine the probability of $X = a$, summing over all possible committee compositions weighted by their likelihood under current settings of N, p, and f_M:

$$\Pr(X = a) = \sum_{i=0}^{H} \sum_{j=0}^{M} \Pr(X = a \mid B_H = i, B_M = j) \Pr(B_H = i) \Pr(B_M = j) \quad (4.1)$$

4.1.2.2 Choosing Quorum

To find an optimal p and Q for different network and model settings, different conditions must be met for the system to be secure. These conditions must hold with respect to a chosen security parameter b, meaning their failure probabilities are bounded by 2^{-b}. The protocol is considered to fail if an adversarial condition is satisfied, causing a security failure, or if the number of active participants in the committee is insufficient to reach the quorum threshold, causing a liveness failure. These two failure events are modelled separately, as either a PMF when the failure probability is expressed as an equality, or a CDF when it is expressed as an inequality.

Depending on the network settings and the requirements for the quorum in the protocol, these conditions can differ.

When assuming reliable broadcast and a protocol requiring quorums with **"at least one honest party"**, the conditions are:

$$\Pr(X = Q) \text{ and } \Pr(B_N < Q), \quad \text{where } X \sim \text{Hyper}(B_N, B_M, Q).$$

The first condition is the security condition for the *at least one honest* case, ensuring that the quorum contains at least one honest participant, which prevents the adversary from forming a fully malicious quorum. This is described by the PMF in Equation 4.1 with $a = Q$. The second condition ensures liveness, requiring that there are enough participants in the committee to reach the quorum threshold Q.

Under the **"honest majority"** requirement, the conditions are:

$$\Pr\left(X \geq \left\lfloor \frac{Q}{2} \right\rfloor\right) \text{ and } \Pr(B_N < Q) \quad \text{where } X \sim \text{Hyper}(B_N, B_M, Q).$$

The first condition ensures that Q is large enough so that any randomly sampled quorum of size Q from the committee does not contain a malicious majority. The second is the same liveness as before.

In an asynchronous network, the conditions differ from those under reliable broadcast because the analysis must account for shadow subcommittees. Previously, the

analysis assumed that a quorum of size Q is drawn from the committee, with each member being randomly honest or malicious. The asynchronous setting additionally assumes Byzantine behaviour, where a malicious Channel Maintainer can withhold or delay honest votes.

Under the **"at least one honest party"** requirement, the conditions are:

$$\Pr(Y = Q) \text{ and } \Pr(B_N < Q), \quad \text{where } Y \sim \text{Hyper}(B_N, B_{H/2} + B_M, Q),$$

with $B_{H/2} \sim \text{Bin}(H/2, p)$. The security condition corresponds to the probability that a randomly chosen quorum is composed entirely of malicious participants together with honest participants all voting for the same state. As discussed in Section 3.6.2, the adversary's optimal strategy distributes honest votes evenly between two states, which is why $H/2$ appears in the distribution. The liveness condition is unchanged from the reliable broadcast assumption.

Under the **"honest majority"** requirement, the conditions are:

$$\Pr(Y = Q), \quad \Pr\left(X > \left\lfloor \frac{Q}{2} \right\rfloor\right), \quad \text{and} \quad \Pr(B_N < Q),$$

where

$$Y \sim \text{Hyper}(B_N, B_{H/2} + B_M, Q), \quad X \sim \text{Hyper}(B_N, B_M, Q).$$

The first condition ensures that the probability of forming shadow subcommittees is negligible, the second ensures that no quorum of size Q contains a malicious majority, and the third is the same liveness condition as before.

4.1.3 General quorum probability identities

General identities are derived for the probabilities of different events when sampling from a randomly formed committee. In this section, the term *nodes* is used instead of *participants* for the sake of generality.

Consider a population of N nodes partitioned into two groups:

$$N = F + G,$$

where F denotes the size of a "bad" group, for example faulty or adversarial nodes, and G denotes the size of a "good" group. The generic terms "bad" and "good" are used here because the identities derived in this section are later applied to several different group definitions. For example, $F = M$ under the reliable broadcast assumption and $F = M + H/2$ under the asynchronous network assumption (Section 4.1.2.2).

Each node is independently selected into a committee with probability p. Let

$$B_F \sim \text{Bin}(F, p), \qquad B_G \sim \text{Bin}(G, p),$$

denote the number of selected nodes from each group. The total committee size is

$$B_N \sim B_F + B_G \qquad \text{where } B_N \sim \text{Bin}(N, p).$$

Thus B_N represents the size of the randomly formed committee.

After the committee is formed, a subset of size a is sampled uniformly at random from the committee, provided that $B_N \geq a$. This sampled subset models a quorum or decision group.

Let $X =$ number of selected nodes from the bad group within the sample of size a.

X then follows a hypergeometric distribution.

The quantities of interest are probabilities of adverse events, such as:

- $X = a$ (all sampled nodes are from the bad group), or
- $X \geq b$ (at least b sampled nodes are from the bad group),

where $0 \leq b \leq a$.

4.1.3.1 Exact probability of total failure

Conditioning on the total committee size

$$B_N \sim B_F + B_G.$$

gives

$$\Pr(X = a) = \sum_{k=0}^{N} \Pr(B_N = k) \Pr(X = a \mid B_N = k).$$

The quantity to study is therefore

$$\Pr(X = a \mid B_N = k).$$

This can be further expanded by conditioning on the number of bad nodes in the committee:

$$\Pr(X = a \mid B_N = k) = \sum_{j} \Pr(X = a \mid B_N = k, B_F = j) \Pr(B_F = j \mid B_N = k),$$

where j are the bad nodes in the committee given that the committee has size k.

If there are j bad nodes, the probability that all a sampled nodes are bad is

$$\frac{\binom{j}{a}}{\binom{k}{a}}.$$

Summing over all possible j gives

$$\sum_j \Pr(X = a \mid B_N = k, B_F = j) \Pr(B_F = j \mid B_N = k) = \sum_{j=0}^{k} \frac{\binom{j}{a}}{\binom{k}{a}} \Pr(B_F = j \mid B_N = k).$$

Conditioned on $B_N = k$, the committee is a uniformly random k-subset of the population, so

$$B_F \mid B_N = k \sim \text{Hyper}(N, F, k),$$

and hence

$$\Pr(B_F = j \mid B_N = k) = \frac{\binom{F}{j}\binom{G}{k-j}}{\binom{N}{k}}.$$

Therefore,

$$\Pr(X = a \mid B_N = k) = \sum_{j=0}^{k} \frac{\binom{j}{a}}{\binom{k}{a}} \frac{\binom{F}{j}\binom{G}{k-j}}{\binom{N}{k}} = \frac{1}{\binom{k}{a}\binom{N}{k}} \sum_{j=0}^{k} \binom{j}{a}\binom{F}{j}\binom{G}{k-j}.$$

Using the identity

$$\binom{F}{j}\binom{j}{a} = \binom{F}{a}\binom{F-a}{j-a},$$

and noting that $\binom{j}{a} = 0$ for $j < a$, the sum can start at $j = a$, giving

$$\Pr(X = a \mid B_N = k) = \frac{\binom{F}{a}}{\binom{k}{a}\binom{N}{k}} \sum_{j=a}^{k} \binom{F-a}{j-a}\binom{G}{k-j}.$$

Now let

$$r = j - a.$$

Then

$$\sum_{j=a}^{k} \binom{F-a}{j-a}\binom{G}{k-j} = \sum_{r=0}^{k-a} \binom{F-a}{r}\binom{G}{k-a-r}.$$

By Vandermonde's identity [18, p. 3],

$$\sum_{r=0}^{k-a} \binom{F-a}{r}\binom{G}{k-a-r} = \binom{N-a}{k-a}.$$

Thus

$$\Pr(X = a \mid B_N = k) = \frac{\binom{F}{a}\binom{N-a}{k-a}}{\binom{k}{a}\binom{N}{k}}.$$

Finally, using

$$\binom{N}{k}\binom{k}{a} = \binom{N}{a}\binom{N-a}{k-a},$$

it follows that

$$\Pr(X = a \mid B_N = k) = \frac{\binom{F}{a}}{\binom{N}{a}} \qquad \text{for } k \geq a,$$

and

$$\Pr(X = a \mid B_N = k) = 0 \qquad \text{for } k < a.$$

Hence

$$\Pr(X = a) = \sum_{k=a}^{N} \Pr(B_N = k) \frac{\binom{F}{a}}{\binom{N}{a}} = \Pr(B_N \geq a) \frac{\binom{F}{a}}{\binom{N}{a}}.$$

The final expression reveals that the conditional probability $\Pr(X = a \mid B_N = k)$ is the same for every $k \geq a$, so the sum over all committee sizes collapses to $\Pr(B_N \geq a)$ multiplied by a single hypergeometric factor, the probability of drawing a bad nodes directly from the full population of N nodes.

4.1.3.2 General failure threshold

The same calculation now generalises to the event that at least b of the a sampled nodes are from the bad group, where $0 \leq b \leq a$.

Conditioning on B_N,

$$\Pr(X \geq b) = \sum_{k=0}^{N} \Pr(B_N = k) \Pr(X \geq b \mid B_N = k).$$

The first quantity to consider is

$$\Pr(X \geq b \mid B_N = k).$$

Given a committee of size k, suppose it contains j bad nodes. If exactly i of the sampled a nodes are bad, the sample must contain

- i nodes from the j bad committee members, and
- $a - i$ nodes from the $k - j$ good committee members.

The total number of size-a samples from the committee is $\binom{k}{a}$. Therefore,

$$\Pr(X = i \mid B_F = j,\ B_N = k) = \frac{\binom{j}{i}\binom{k-j}{a-i}}{\binom{k}{a}}.$$

Summing over all $i \geq b$, gives

$$\Pr(X \geq b \mid B_F = j,\ B_N = k) = \sum_{i=b}^{a} \frac{\binom{j}{i}\binom{k-j}{a-i}}{\binom{k}{a}}.$$

Now average over all possible values of j:

$$\Pr(X \geq b \mid B_N = k) = \sum_{j=0}^{k} \left(\sum_{i=b}^{a} \frac{\binom{j}{i}\binom{k-j}{a-i}}{\binom{k}{a}} \right) \Pr(B_F = j \mid B_N = k).$$

Since

$$\Pr(B_F = j \mid B_N = k) = \frac{\binom{F}{j}\binom{G}{k-j}}{\binom{N}{k}},$$

this gives

$$\Pr(X \geq b \mid B_N = k) = \sum_{j=0}^{k} \left(\sum_{i=b}^{a} \frac{\binom{j}{i}\binom{k-j}{a-i}}{\binom{k}{a}} \right) \frac{\binom{F}{j}\binom{G}{k-j}}{\binom{N}{k}}.$$

Interchanging the order of summation gives

$$\Pr(X \geq b \mid B_N = k) = \frac{1}{\binom{k}{a}\binom{N}{k}} \sum_{i=b}^{a} \sum_{j=0}^{k} \binom{j}{i}\binom{k-j}{a-i}\binom{F}{j}\binom{G}{k-j}.$$

Now fix i. Since $\binom{j}{i} = 0$ for $j < i$, and $\binom{k-j}{a-i} = 0$ unless $j \leq k - a + i$, the inner sum reduces to

$$\sum_{j=i}^{k-a+i} \binom{j}{i}\binom{k-j}{a-i}\binom{F}{j}\binom{G}{k-j}.$$

The identities

$$\binom{F}{j}\binom{j}{i} = \binom{F}{i}\binom{F-i}{j-i},$$

and

$$\binom{G}{k-j}\binom{k-j}{a-i} = \binom{G}{a-i}\binom{G-(a-i)}{k-j-(a-i)}.$$

are now applied.

Thus the inner sum becomes

$$\binom{F}{i}\binom{G}{a-i} \sum_{j=i}^{k-a+i} \binom{F-i}{j-i}\binom{G-(a-i)}{k-j-(a-i)}.$$

Now set

$$r = j - i.$$

Then

$$j = i + r, \qquad k - j - (a - i) = k - a - r,$$

and the sum becomes

$$\binom{F}{i}\binom{G}{a-i} \sum_{r=0}^{k-a} \binom{F-i}{r}\binom{G-(a-i)}{k-a-r}.$$

Applying Vandermonde's identity,

$$\sum_{r=0}^{k-a} \binom{F-i}{r}\binom{G-(a-i)}{k-a-r} = \binom{N-a}{k-a}.$$

Therefore,

$$\Pr(X \geq b \mid B_N = k) = \frac{1}{\binom{k}{a}\binom{N}{k}} \sum_{i=b}^{a} \binom{F}{i}\binom{G}{a-i}\binom{N-a}{k-a}.$$

Factoring out terms independent of i,

$$\Pr(X \geq b \mid B_N = k) = \frac{\binom{N-a}{k-a}}{\binom{k}{a}\binom{N}{k}} \sum_{i=b}^{a} \binom{F}{i}\binom{G}{a-i}.$$

Using again

$$\binom{N}{k}\binom{k}{a} = \binom{N}{a}\binom{N-a}{k-a},$$

it follows that for $k \geq a$,

$$\Pr(X \geq b \mid B_N = k) = \sum_{i=b}^{a} \frac{\binom{F}{i}\binom{G}{a-i}}{\binom{N}{a}},$$

and for $k < a$,

$$\Pr(X \geq b \mid B_N = k) = 0.$$

Hence

$$\Pr(X \geq b) = \sum_{k=a}^{N} \Pr(B_N = k) \sum_{i=b}^{a} \frac{\binom{F}{i}\binom{G}{a-i}}{\binom{N}{a}} = \Pr(B_N \geq a) \sum_{i=b}^{a} \frac{\binom{F}{i}\binom{G}{a-i}}{\binom{N}{a}}.$$

The final expression reveals that the conditional probability $\Pr(X \geq b \mid B_N = k)$ is the same for every $k \geq a$, so the sum over all committee sizes collapses to $\Pr(B_N \geq a)$ multiplied by a sum of hypergeometric factors over all values $i \geq b$, each giving the probability of drawing exactly i bad nodes in a size-a sample directly from the full population of N nodes.

4.1.4 Finding set size from Channel Maintainer

To analyse a setting where the Channel Maintainer has the power to selectively forward or withhold messages, as in the asynchronous network setting described in Section 3.1.4, a new parameter S is introduced. S denotes the minimum number of votes that a participant needs to receive from the Channel Maintainer before accepting a state. The quorum of size Q is then drawn from this set of S received votes, rather than from the whole committee. Since the Channel Maintainer is assumed to be malicious, it is assumed that the S received votes contain all malicious votes from the committee. The remaining part of S is then filled by honest votes. This introduces additional security and liveness conditions on S, presented in this section.

Recall from Section 3.5.3 that liveness requires honest participants to be able to accept new states. Two conditions on S follow: S cannot exceed the committee size, and the malicious participants alone must not be able to fill the entire set of S. Violating either condition would prevent honest votes from contributing, causing the participant to fail to accept any state. These two conditions are expressed as

$$\Pr(B_N < S) < 2^{-b}, \quad \Pr(B_M > S) < 2^{-b}. \tag{4.2}$$

Since the Channel Maintainer is assumed to be malicious, the set of size S is assumed to contain all malicious committee members, with the remaining slots filled by honest votes. Under this worst-case assumption, the security condition for total failure is

$$\frac{\binom{B_F}{Q}}{\binom{S}{Q}},$$

where $F = M$ for reliable broadcast and $F = M + H/2$ for asynchronous network.

And for a general threshold failure the condition is

$$\sum_{i=b}^{a} \frac{\binom{B_F}{i}\binom{S-B_F}{a-i}}{\binom{S}{a}},$$

where $b = \lfloor Q/2 \rfloor$, $a = Q$, and $F = M$ in the honest majority setting.

With this method, *reliable broadcast* is no longer guaranteed. What is referred to as *reliable broadcast* is instead a *semi-reliable broadcast*, where every committee member broadcasts their vote to the Channel Maintainer, who selectively forwards a subset of these messages to the participants of the protocol, broadcasting the same set of size S to all of them.

4.1.4.1 Security for a set of size S

A set of size S drawn from B_N is considered, from which a subset of size a is sampled uniformly at random. In both cases below, all faulty nodes are assumed to be contained within the set of size S, representing the worst-case scenario.

General total failure Since the exact number of faulty nodes is unknown, the security condition can be written as

$$\frac{\binom{B_F}{a}}{\binom{S}{a}} < 2^{-b}.$$

This can more precisely be expressed as

$$\sum_{j=0}^{S} \frac{\binom{j}{a}}{\binom{S}{a}} \Pr(B_F = j) + \sum_{j=S+1}^{F} \frac{\binom{S}{a}}{\binom{S}{a}} \Pr(B_F = j),$$

which, by substituting the binomial distribution $B_F \sim \mathrm{Bin}(F, p)$, simplifies to

$$\frac{1}{\binom{S}{a}} \sum_{j=0}^{S} \binom{j}{a}\binom{F}{j} p^j (1-p)^{F-j} + \Pr(B_F > S).$$

Applying the identity

$$\binom{F}{j}\binom{j}{a} = \binom{F}{a}\binom{F-a}{j-a}$$

and substituting $k = j - a$ gives

$$\frac{\binom{F}{a}}{\binom{S}{a}} p^a \sum_{k=0}^{S-a} \binom{F-a}{k} p^k (1-p)^{(F-a)-k} + \Pr(B_F > S).$$

The sum is the CDF of a binomial distribution, so the expression becomes

$$\Pr(B_F > S) + \frac{\binom{F}{a}}{\binom{S}{a}} p^a \Pr(B_{F-a} \leq S - a), \qquad B_{F-a} \sim \mathrm{Bin}(F-a, p). \tag{4.3}$$

General threshold failure For a general threshold $0 \leq b \leq a$, the probability that at least b of the a sampled nodes are bad is

$$\sum_{i=b}^{a} \frac{\binom{B_F}{i}\binom{S-B_F}{a-i}}{\binom{S}{a}}.$$

Conditioning on the value of B_F and separating the cases $B_F \leq S$ and $B_F > S$ gives

$$\sum_{i=b}^{a} \sum_{j=0}^{S} \frac{\binom{j}{i}\binom{S-j}{a-i}}{\binom{S}{a}} \Pr(B_F = j) + \sum_{i=b}^{a} \sum_{j=S+1}^{F} \frac{\binom{S}{i}\binom{0}{a-i}}{\binom{S}{a}} \Pr(B_F = j).$$

Since $\binom{0}{a-i} = 0$ unless $a = i$, the right double sum reduces to $\Pr(B_F > S)$, giving

$$\sum_{i=b}^{a} \sum_{j=0}^{S} \frac{\binom{j}{i}\binom{S-j}{a-i}}{\binom{S}{a}} \Pr(B_F = j) + \Pr(B_F > S). \tag{4.4}$$

4.2 Computational Implementation

In this section the implementation in both Python and R is explained.

4.2.1 Optimal p and Q

This section describes the implementation for finding the smallest expected committee size $C = pN$ and corresponding Q such that both the security and liveness failure probabilities are bounded by 2^{-b}, across all four protocol settings.

4.2.1.1 Arithmetic Precision

The failure probabilities targeted in this analysis can be as small as 2^{-128}. Standard floating-point numbers in Python (`float`) do not have enough precision to represent values this small accurately. To handle them, all probability computations in the Python scripts are performed using the `gmpy2` library (Python 3.10), which provides arbitrary-precision floating-point arithmetic through its `mpfr` type. The working precision is set to 256 bits by calling `gmpy2.get_context().precision = 256` at the start of each script. The R implementation uses the `Rmpfr` package, which

uses the same underlying MPFR library, with the same 256-bit precision set with `precBits = 256L`.

All values produced during the computation, such as selection probabilities, binomial CDF terms, and combinatorial ratios, are stored as arbitrary-precision floating-point values throughout.

4.2.1.2 Shared computational building blocks

All four Python scripts and the R script share a set of core functions that the higher-level failure probability computations build on.

The function `binom_cdf_leq(q, n, p)` (Listing A.1) computes the binomial CDF $\Pr(B_N \leq q)$. It starts by computing the $k = 0$ term and then iteratively derives each next term from the previous one by multiplying by a single factor, adding each term to a running sum. This avoids computing binomial coefficients independently for each term. The upper tail $\Pr(B_N \geq Q)$, used in both the security and liveness computations, is obtained by the function `binom_cdf_geq(Q, N, p)` (Listing A.2) as one minus the CDF evaluated at $Q - 1$. In the R script, the `pbetaI` function from the `Rmpfr` package is used instead to compute the CDF in a single call, which evaluates the regularised incomplete beta function.

The function `comb_div(F, N, Q)` (Listing A.3) computes the ratio:

$$\frac{\binom{F}{Q}}{\binom{N}{Q}}$$

that appears in the closed-form identities derived in Section 4.1.3.1. It evaluates the ratio as a product of Q individual fractions, each computed as a division of two `mpfr` values, instead of computing the two binomial coefficients separately.

4.2.1.3 Security Failure Probability

Depending on the security condition, the security probability is computed differently.

For the *at least one honest* condition, the implementation uses the closed-form identity derived in Section 4.1.3.1. The function `prob_X_eq_Q(N, M, p, Q)` (Listing A.4) computes $\Pr(X = Q)$ as the product of `binom_cdf_geq` and `comb_div`. The only difference between reliable broadcast and asynchronous network is the adversarial population size in `comb_div`. The script for reliable broadcast uses the number of malicious participants M, while the asynchronous network script instead uses $M + \lfloor H/2 \rfloor$, to account for the shadow subcommittees described in Section 3.6.

Under the *honest majority* condition, the closed-form simplification cannot be used. The function `hypergeom_tail_population_gmp(N, K, Q, q)` (Listing A.5) instead computes the hypergeometric tail $\Pr(X \geq \lfloor Q/2 \rfloor)$ directly. It uses `gmpy2.bincoef` to evaluate the first term and then obtains each following term by multiplying by a

single factor, in the same iterative style as the binomial CDF function. The product of this hypergeometric tail and $\Pr(B_N \geq Q)$ is used as the security probability.

The Python script for an asynchronous network with *honest majority* evaluates three failure conditions per quorum Q instead of two. It computes an async split failure using the same closed-form identity as the *at least one honest* script but with $F = M + \lceil H/2 \rceil$, in addition to the *honest majority* failure and the liveness failure. The maximum of all three is the failure probability for a given Q.

4.2.1.4 Quorum selection

For a fixed expected committee size n and selection probability $p = n/N$, the implementation finds the quorum Q that minimises the overall failure probability while keeping all failure conditions below 2^{-b}.

All scripts start by using a binary search over Q to find the crossing point where the security failure probability drops below the liveness failure probability. As Q increases, the security probability decreases while the liveness probability increases, so this crossing always exists within the search range.

After the crossing point is found, the scripts evaluate a range of Q values. For the *at least one honest* scripts, this range is small, covering three values on each side of the crossing. For the *honest majority* scripts, the range is larger and expands in stages if no valid Q is found. Starting from a base size of 64 on each side of the crossing, it doubles to 128, then to 256, until a valid Q is found. Each Q value within this range is evaluated and the one with the lowest maximum failure probability below 2^{-b} is selected.

To parallelise, the *honest majority* scripts use Python's `multiprocessing` module. The Q values are split into chunks and sent to separate worker processes through a process pool. Each worker computes the failure probabilities for its assigned Q values and returns the best one. To then find the overall best Q, the results from all workers are compared. The number of workers is set to the number of available CPU cores.

4.2.1.5 Parameter search over committee size

The outer layer of the implementation searches for the smallest expected committee size n that gives a valid quorum Q for the chosen bit security level b. The function `find_best_n` (Listing A.6) handles this. It iterates over a grid of n values from smallest to largest. For each n, the selection probability is set to $p = n/N$ and the quorum selection described in the previous subsection is applied. The first n for which a valid Q is found is returned, along with the corresponding Q, failure probabilities, and $\log_2$ of the overall failure probability.

4.2.1.6 Cross-validation in R

The reliable broadcast *at least one honest* case is implemented in R using the `Rmpfr` package, to verify the Python results. The R script also uses the same closed-form identity for the security failure probability, the same binary search for the crossing point, and the same search over a grid of n values, as the Python scripts described in the previous subsections. The core difference is in how the binomial CDF is computed. Instead of building the CDF iteratively term by term as the Python scripts do, the R script computes it in a single call using the `pbetaI` function from `Rmpfr` (Listing A.12).

A comparison between the results from the implementations in both R and Python is then done to confirm whether they produce the same optimal Q and failure probabilities for each parameter combination.

4.2.2 Optimal S

There are several ways to construct a set of size S that satisfies the required conditions. Initially, the results from Section 4.2.1 were used as fixed parameters, and S was determined accordingly. Later, an alternative approach was implemented in which p is fixed, and the goal is instead to find the largest S satisfying the liveness requirement, together with the smallest Q satisfying the security requirement for that S.

4.2.2.1 Finding S for fixed p and Q

The implementation uses the `gmpy2` library with 128-bit floating-point precision to avoid numerical underflow when evaluating probabilities on the order of 2^{-b}.

Given fixed values of p and Q, each candidate value of S is evaluated against two conditions. The security probability $\Pr_{\text{sec}}(S)$ is computed depending on the setting: either using Equation 4.4 for the *honest majority* case, with $F = M$, or Equation 4.3 for the *at least one honest* case, with $F = M$ for reliable broadcast and $F = M+H/2$ for the asynchronous network.

For the *at least one honest* case, the binomial coefficient ratio is computed by the function `binom_ratio` (Listing A.7), which evaluates it as a product of Q terms of the form $(F - t)/(S - t)$ for $t = 0, \ldots, Q - 1$, where the size F depends on the network setting. The binomial CDF is evaluated using `binom_cdf_leq`, which relies on a recurrence relation for the binomial PMF, while its complement is provided by `binom_cdf_geq`.

In the *honest majority* case, the security probability is computed by the function `security_sum` (Listing A.9). This function first constructs the full PMF of $\text{Bin}(M, p)$ for $j = 0, \ldots, S$ using a recurrence relation, where each term $\Pr(\text{Bin}(M, p) = j)$ is derived from the previous one. The tail probability $\Pr(\text{Bin}(M, p) > S)$ is then obtained as one minus the cumulative sum. A double sum is evaluated by iterating

i from $\lfloor Q/2 \rfloor$ to Q, and for each i, iterating j over the valid range $j \in [i, S - Q + i]$. For each pair (i, j), the result is added to the total using `binom_gmp` (Listing A.8) for the binomial coefficients. Finally, the sum is divided by the denominator and the tail probability is added. A linear scan over all values identifies S_{sec}, the smallest S such that $\Pr_{\text{sec}}(S) < 2^{-b}$.

In the case of *honest majority* in an asynchronous network an additional function was used called `max_prob` (Listing A.11) which takes the maximum probability of `security_sum` and the *at least one honest* case with $F = M + H/2$.

The liveness probability is evaluated using `binom_cdf_leq`, and a linear scan determines S_{liv}, the largest S for which $\Pr_{\text{liv}}(S) < 2^{-b}$.

All values of S in the range $[0, N \cdot p]$ are evaluated, since choosing S larger than the expected committee size $N \cdot p$ would make it unlikely that enough votes arrive to reach S, violating liveness. The results are plotted on a $\log_2$ scale together with the threshold 2^{-b}. If $S_{\text{sec}} \leq S_{\text{liv}}$, then the interval $[S_{\text{sec}}, S_{\text{liv}}]$ defines a feasible region where any integer S satisfies both constraints. Otherwise, no such S exists for the given parameters.

4.2.2.2 Finding the smallest Q for given p

An alternative approach fixes p and determines S and Q jointly. The function `find_S` (Listing A.10) searches downward from $\lfloor p \cdot N \rfloor$ to find the largest S satisfying both the liveness requirement and a security requirement

$$\Pr(B_N < S) < 2^{-b}, \quad \Pr(B_M > S) < 2^{-b}. \tag{4.5}$$

Specifically, it checks that $P(\text{Bin}(M, p) > S) < 2^{-b}$ and $P(\text{Bin}(N, p) < S) < 2^{-b}$ using `pbinom_gmp_gt` and `binom_cdf_leq`, respectively. If no such S exists, the procedure terminates.

Given this value of S, the security probability is evaluated for all quorum sizes $Q \in [1, S)$. In the *at least one honest* model, $\Pr_{\text{sec}}(Q)$ is computed using Equation 4.3, with $F = M$ for reliable broadcast and $F = M + H/2$ for the asynchronous setting, using the same method as above.

In the *honest majority* model with reliable broadcast, $\Pr_{\text{sec}}(Q)$ is computed using Equation 4.4, with $F = M$, via the `security_sum` function. In the asynchronous network $\Pr_{\text{sec}}(Q)$ is the `max_prob` function, which is the same as described in section 4.2.2.1.

In both cases, a linear scan identifies Q_{sec}, the smallest Q such that $\Pr_{\text{sec}}(Q) < 2^{-b}$. The results are plotted on a $\log_2$ scale together with the threshold 2^{-b}.

5
Results

This chapter presents the results of the probabilistic analysis and computational implementation described in the previous chapter. First, the optimal values of p and Q are presented for each network setting and security condition, and compared against Tail-Hammer [10]. Then, the results for the set size S are presented across the different settings.

5.1 Choosing optimal p and Q

Table 5.1 presents the optimal values of the selection probability p and quorum size Q across different settings. These values are computed to satisfy respective security and liveness constraints while minimising quorum size to improve efficiency in a system where all endorsements from committee members are assumed to reach each participant in the protocol. Variables chosen for the computation of the results are the same as in Tail-Hammer to be able to compare results [10]. The results from Tail-Hammer can be found in Table 2.1.

Table 5.1: Optimal p and Q computed with the technique discussed in Section 4.2.1.

At least one honest party case

		Reliable Broadcast			Asynchronous Network		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	**249**	**249**	**247**	**454**	**453**	**446**
	Q	**74**	**74**	**74**	**206**	**206**	**204**
60	C	**115**	**115**	**115**	**211**	**211**	**208**
	Q	**35**	**35**	**35**	**97**	**97**	**96**

Honest majority case

		Reliable Broadcast			Asynchronous Network		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	**1452**	**1434**	**1306**	**1452**	**1434**	**1306**
	Q	**984**	**972**	**888**	**984**	**972**	**888**
60	C	**658**	**655**	**627**	**658**	**655**	**627**
	Q	**446**	**444**	**426**	**446**	**444**	**426**

5.1.1 Comparing with R

A script in R was implemented, making the same computations as in the Python script, using the high-precision R library called `Rmpfr`. This leads to the same results of the optimal p and Q as in Python, which validates the results from Python.

Table 5.2: Optimal p and Q computed with the technique discussed in Section 4.2.1.6.

At least one honest party case

		R Script			Python Script		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	**249**	**249**	**247**	249	249	247
	Q	**74**	**74**	**74**	74	74	74
60	C	**115**	**115**	**115**	115	115	115
	Q	**35**	**35**	**35**	35	35	35

5.2 Finding S, Q, and p for unreliable central server

The results in Table 5.3 show the smallest values of S required to satisfy the security condition when using the parameter values from Table 5.1. Some entries in the table are missing due to the high computational complexity of the analysis. Figures 5.1-5.3 show the plot for selecting S with values p and Q from Table 5.1, where the *honest majority* case for both reliable broadcast and asynchronous network is represented in Figure 5.3. It highlights the tension when choosing between the security and liveness requirements when selecting the set size S.

Furthermore, Figures 5.4- 5.6 are the results from taking p from Tail-Hammer [10], getting largest the S that fulfils the liveness and security requirement from Equation 4.2, and then selecting a Q using either Equation 4.3 or 4.4 depending on the requirement. Figure 5.7 shows the relationship when choosing the largest S and smallest Q for increasing values of p on one population composition with $b = 60$.

Table 5.3: Results for secure set size S where $C = N \cdot p$ and Q. Entries marked **na** were not computed due to runtime constraints.

At least one honest party

		Reliable Broadcast			Asynchronous Network		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	249	249	247	454	453	446
	Q	74	74	74	206	206	204
	S	**286**	**285**	**281**	**561**	**559**	**547**
60	C	115	115	115	211	211	208
	Q	35	35	35	97	97	96
	S	**131**	**131**	**131**	**261**	**261**	**257**

Honest majority

		Reliable Broadcast			Asynchronous Network		
b	N	10^9	10^5	10^4	10^9	10^5	10^4
128	C	1452	1434	1306	1452	1434	1306
	Q	984	972	888	984	972	888
	S	**na**	**na**	**1420**	**na**	**na**	**1420**
60	C	658	655	627	658	655	627
	Q	446	444	426	446	444	426
	S	**719**	**716**	**684**	**719**	**716**	**684**

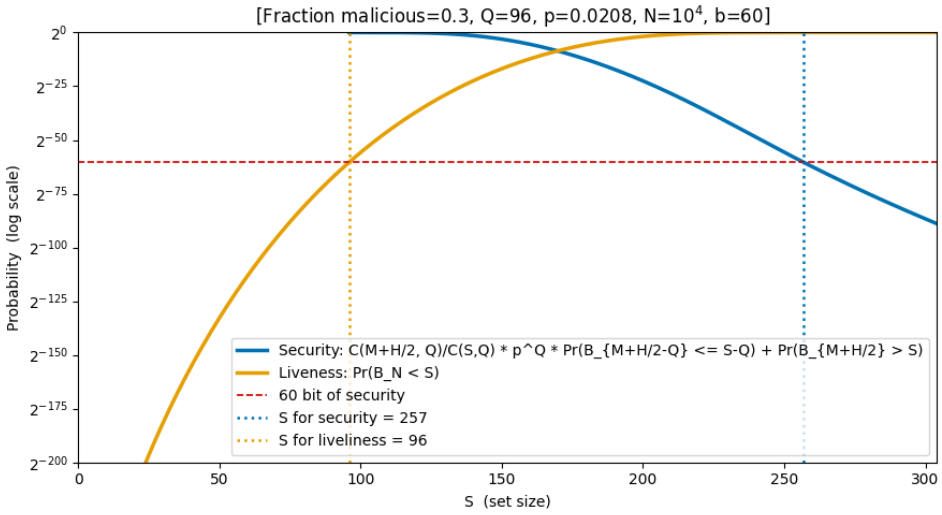

Figure 5.1: *At least one honest* party in asynchronous network where security is $\frac{\binom{B_{M+H/2}}{Q}}{\binom{S}{Q}}$ and liveness $\Pr(S > B_N)$. The red dashed line is the bit security level while the yellow and blue dotted lines are the required S for liveness and security respectively. The value $p = \frac{C}{N}$ and the values C and Q are taken from Table 5.1.

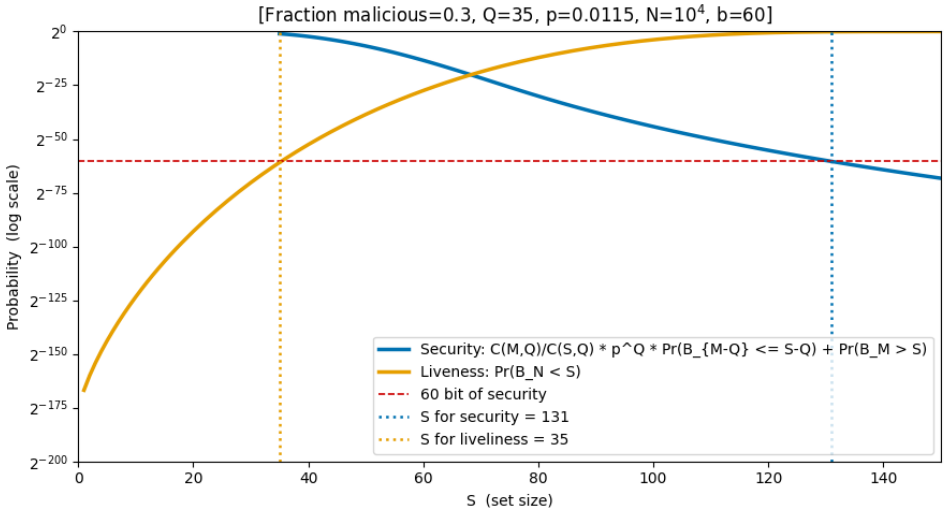

Figure 5.2: *At least one honest* party in reliable broadcast where security is $\frac{\binom{B_M}{Q}}{\binom{S}{Q}}$ and liveness $\Pr(S > B_N)$. The red dashed line is the bit security level while the yellow and blue dotted lines are the required S for liveness and security respectively. The value $p = \frac{C}{N}$ and the values C and Q are taken from Table 5.1

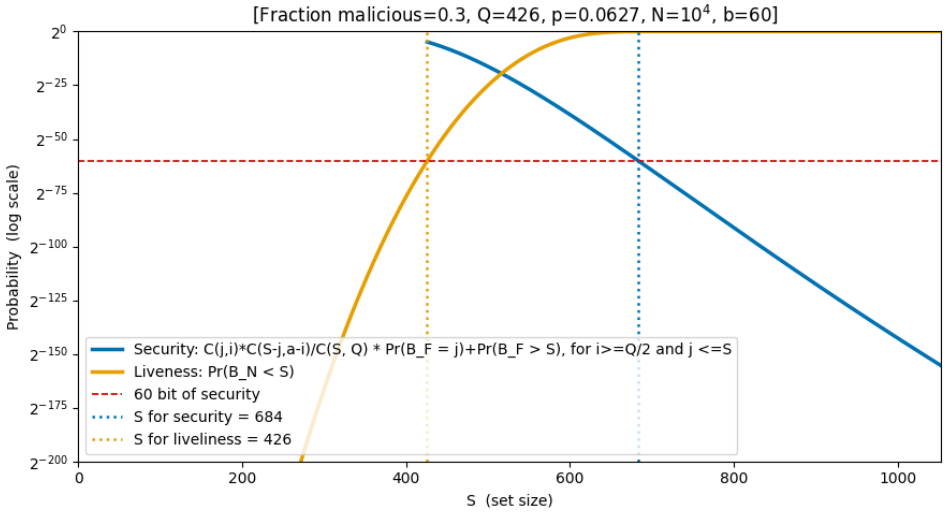

Figure 5.3: *Honest majority* in both network settings where security is $\sum_{q=\lfloor Q/2 \rfloor}^{Q} \frac{\binom{B_M}{q}\binom{S-B_M}{Q-q}}{\binom{S}{Q}}$ and liveness $\Pr(S > B_N)$. The red dashed line is the bit security level while the yellow and blue dotted lines are the required S for liveness and security respectively. The value $p = \frac{C}{N}$ and the values C and Q are taken from Table 5.1.

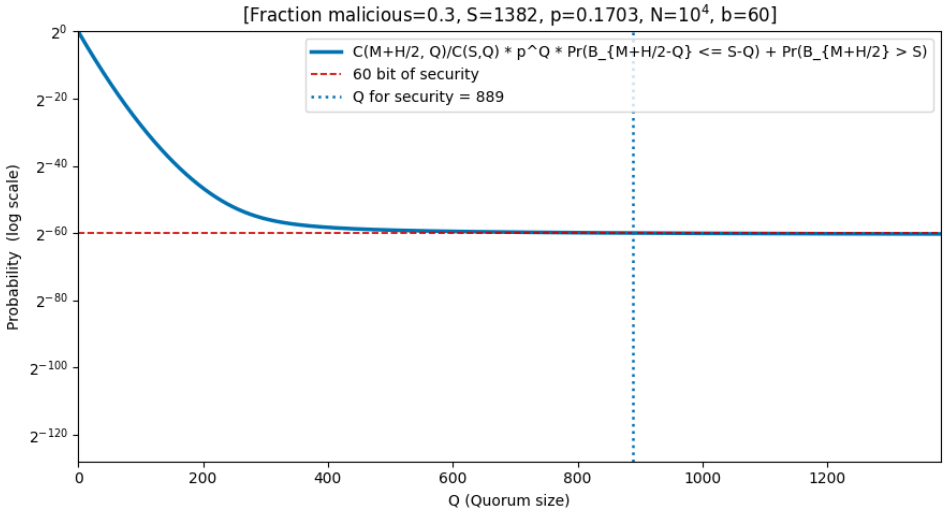

Figure 5.4: Probability of *at least one honest* party in an asynchronous network, where p is from Tail-Hammer [10]. S is the largest value satisfying both security and liveness requirements. The red dashed line indicates the bit security level and the blue dotted line marks the Q meeting this level.

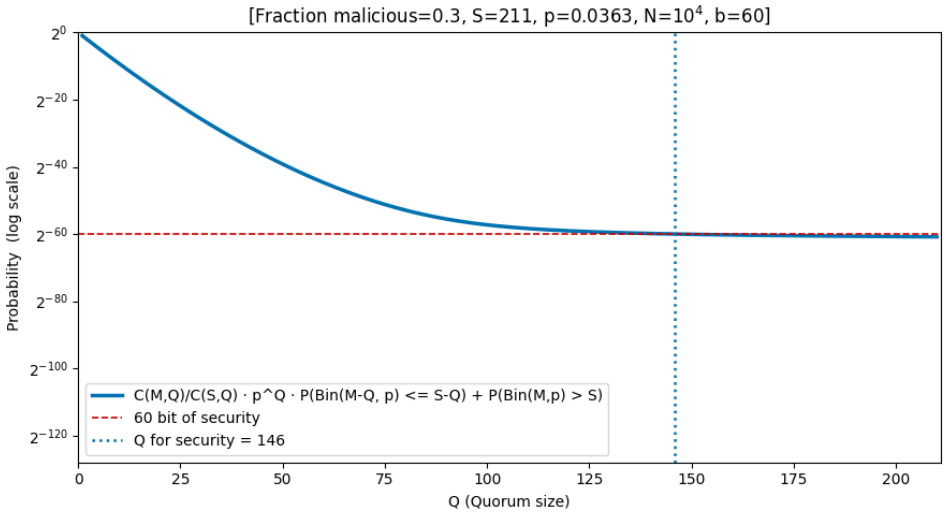

Figure 5.5: Probability of *at least one honest* party in a reliable broadcast, where p is from Tail-Hammer [10]. S is the largest value satisfying both security and liveness requirements. The red dashed line indicates the bit security level and the blue dotted line marks the Q meeting this level.

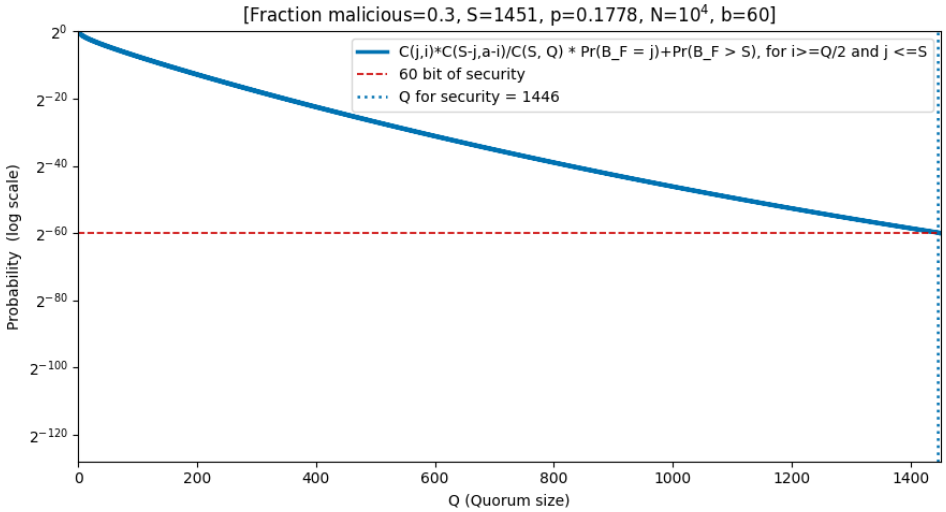

Figure 5.6: Probability of *honest majority* in both network settings, where p is from Tail-Hammer [10]. S is the largest value satisfying both security and liveness requirements. The red dashed line indicates the bit security level and the blue dotted line marks the Q meeting this level.

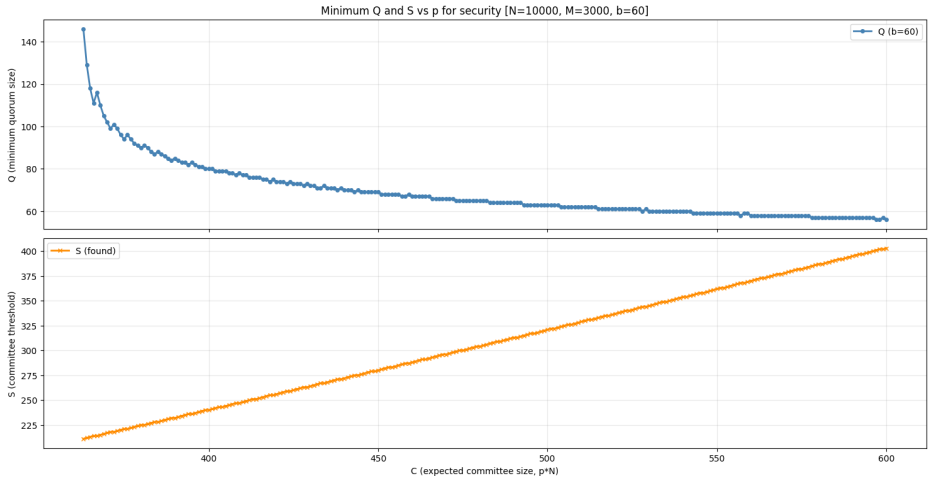

Figure 5.7: *At least one honest* party in a reliable broadcast. Minimum quorum size Q and committee threshold S versus expected committee size $C = p \cdot N$ for $N = 10000$, $M = 3000$, and bit security $b = 60$. The top plot shows that the required Q decreases as the expected committee size grows, while the bottom plot shows that S increases linearly with C.

6
Discussion

This chapter discusses the results and implementation of the proposed framework, as well as directions for future work. We first analyse the obtained parameter choices for p, Q, and S, and analyse the trade-offs between security, liveness, communication cost, and computational overhead in all network settings. We then discuss the implementations in Python and R, including computational challenges and optimisations. Finally, we discuss possible extensions of the work, such as jointly optimising S, p, and Q, and incorporating inactive participants into the model.

6.1 Analysis of result

In this section, we discuss the results from the previous chapter. We first focus on p and Q and then we consider the role of the set size S and its influence on security and liveness in the asynchronous network.

6.1.1 Analysis of result for optimal p and Q

A key observation from Table 5.1 is that the resulting quorum sizes Q are consistently small relative to the expected committee size $C = pN$, while achieving the same b-bit security guarantee. This confirms that the probabilistic committee selection is effective in reducing the number of participants required to reach agreement, thereby lowering both communication and computational overhead.

When comparing these results to the Tail-Hammer framework, we observe a clear improvement with the assumption of getting all votes from the Channel Maintainer. Tail-Hammer already demonstrates that quorum sizes can be chosen significantly smaller than the full committee size by using tighter statistical bounds instead of traditional Chernoff-based estimates [10]. However, the values of Q and p obtained in Table 5.1 are consistently lower than those reported in Tail-Hammer for the same system parameters.

For example, in the *at least one honest* reliable broadcast setting with 128-bit security and a population of $N = 10^9$, Tail-Hammer reports a quorum of size 481 while our approach yields a quorum size of $Q = 74$, a reduction of $\sim 84.6\%$ [10]. This is a significant reduction in the number of votes that each participant needs to cryptographically verify. Another important improvement is the reduction of p, which results in smaller expected committee sizes and therefore reduced bandwidth

usage, since fewer votes are sent in total.

The improvement over Tail-Hammer stems from a difference in how quorum composition is modelled. Tail-Hammer bounds $\Pr(B_M \geq Q)$ or $\Pr(B_{M+H/2} \geq Q)$ directly, implicitly assuming the Channel Maintainer forwards exactly Q votes and the adversary controls which ones. In contrast, our model assumes all committee votes reach each participant, with the quorum drawn uniformly at random without replacement from the full set. The resulting closed-form expression derived in Section 4.1.3 decays much faster than the binomial tail used in Tail-Hammer, since honest members are proportionally represented in any uniformly random sample. It is this tighter bound that allows both Q and $C = pN$ to be smaller while maintaining the same b-bit security level.

The results from Table 5.1 demonstrate that for the *honest majority*, the probability p and quorum size Q are the same for both reliable broadcast and asynchronous network. This points to the *honest majority* security bound being the binding constraint since, for example, the reliable broadcast setting with total population $N = 10^9$ has $C = 1452$ and $Q = 984$ while the asynchronous network setting with the same population size has exactly the same values. It should be noted that this is for a specific fraction of malicious participants $f_M = 0.3$.

However, intuitively, the asynchronous network's security constraint would become the binding bound if the malicious fraction decreases. This follows from the fact that the *honest majority* case checks the risk that the adversary has too many malicious participants available, while the asynchronous constraint checks the risk that the adversary has access to a malicious part plus half of the honest participants. The asynchronous case includes an honest participant term that still works in the adversary's favour and increases as f_M decreases. So as f_M decreases, the *honest majority* constraint becomes less strict, but the asynchronous constraint does so more slowly because half of the (now larger) honest participants are still beneficial for the adversary.

As mentioned previously, this implementation is only valid if we assume all votes from committee members arrive at each participant. So this solution supports a *hybrid asynchronous network*, where committee members can send multiple votes. However, it does not cover the case where a central server, such as the Channel Maintainer in CoD, is corrupted and can selectively forward votes. In such a scenario, a central server could choose to send specific votes to different sets of participants, enabling split-view attacks. To address this, we introduce the set size parameter S, which limits the influence of a malicious central server and helps maintain security.

6.1.2 Analysis of results of set size S

For a fixed choice of p and Q, the security bound decreases as S increases, while the liveness probability $\Pr(S > B_N)$ increases with S. This trade-off is illustrated in Figures 5.1, 5.2, and 5.3, where the blue curve (security) is decreasing and the

green curve (liveness) is increasing. The vertical dashed lines indicate the minimum values of S required to satisfy each constraint individually.

Ideally, one would choose S such that both constraints are satisfied simultaneously. However, for the parameter values (p, Q) used in Figures 5.1 and 5.2, which were obtained from the results in Table 5.1, no such value of S exists. In particular, the value of S required to satisfy the security constraint lies to the right of the value required to satisfy the liveness constraint, meaning that the feasible regions do not overlap.

As a result, the values of S reported in Table 5.3 are chosen to satisfy the security requirement only. These values should therefore be interpreted as lower bounds necessary for security, but they do not guarantee liveness.

This observation highlights an important limitation of the earlier parameter selection procedure. Since p and Q were optimised independently of the set size S, the resulting parameters are not necessarily compatible with the joint feasibility constraints. In particular, the earlier liveness condition $\Pr(Q > B_N) < 2^{-b}$ does not guarantee that a corresponding set size S exists such that $\Pr(S > B_N)$ is also below bound while maintaining security.

Furthermore, although the liveness threshold for S often appears numerically close to Q, this does not imply that a valid solution exists near this region. Instead, the security constraint prohibits choosing S in that range, effectively making the liveness requirement unattainable for the given (p, Q).

One way to address this issue is to modify the parameter selection procedure to ensure joint feasibility. Instead of first selecting (p, Q) based solely on the original security and liveness conditions, one could require that a candidate pair (p, Q) is only accepted if there exists a corresponding set size S that satisfies both the security and liveness constraints.

Such an approach has been implemented, where for each candidate (p, Q) that satisfies the original conditions, an additional check is done to determine whether a feasible S exists. Only pairs for which such an S can be found are accepted. However, this approach is computationally expensive in practice. It requires, for each candidate (p, Q), an additional search over S, resulting in a significantly increased runtime due to nested loops and the use of high-precision arithmetic. As a result, this method is currently not feasible for large-scale parameter searches.

Figures 5.4-5.6 show the security probability as a function of Q, where p is taken from Tail-Hammer and S is the largest value satisfying both security and liveness from Equation 4.2. Since Tail-Hammer computes Q in a similar way to how S is computed, the values are expected to be close. In the reliable broadcast setting, the liveness and security conditions are almost the same in both frameworks, so S and Q are tight. However, in the asynchronous setting, Tail-Hammer uses

$\Pr(B_{M+H/2} \geq Q) < 2^{-b}$ while the security condition for S uses $\Pr(B_M > S) < 2^{-b}$, which is a weaker condition, resulting in a larger S. However, the Q in this analysis is computed using Equation 4.3 with $F = M + H/2$.

The curve flattens around the 2^{-60} security level because once the security condition is well below 2^{-b}, the dominant contribution to the failure probability becomes the liveness tail $\Pr(B_N \geq Q)$, which does not decrease further with Q, causing the curve to stagnate near the bit security level.

In Figure 5.7, a significant trade-off between larger p and smaller Q can be observed. This is when S is the largest set size that fulfils the conditions found in Equation 4.5 for bit security b. The quorum size Q is then the smallest size that fulfils the condition. This shows a significant trade-off between having large committee sizes with larger bandwidth usage and small quorum sizes that reduce the computational load on each participant. If bandwidth usage is a concern, a smaller p can be chosen with a smaller expected committee size. However, that would result in a Q closer to S, and hence increase the computational load on each participant.

6.2 Analysis of implementation

This section discusses the computational implementation in Python and R, including the design choices made to handle the high-precision arithmetic requirements and the challenges encountered during the parameter search.

6.2.1 Finding optimal p and Q

For the *honest majority* setting in both reliable broadcast and asynchronous network models, the Python scripts were slow because they had to sum over all possible numbers of malicious participants from $\lfloor Q/2 \rfloor$ up to Q in order to account for every scenario in which the quorum is at least half malicious. To improve efficiency, the script was parallelised by spawning a set of worker processes, where the number of workers depends on the machine's available CPU cores. Each worker was then assigned a subset of candidate Q values to evaluate. The search starts from the crossing point of the security and liveness tails, which is found with a binary search with time complexity $\mathcal{O}(\log n)$. A window of size 64 of potential Q values is then evaluated, and if a valid Q is found it is returned. If no valid Q is found within this window, it is expanded and the search continues. One limitation of this approach is that some workers are assigned smaller Q values in the lower part of the window while the last worker receives all the larger values, leading to an imbalance of workload.

Before applying the combinatorial identities, we had a double nested loop over the sum of honest and malicious participants respectively as seen in Equation 4.1. The loop then sums over all possible compositions of committees and in every iteration computes the three probabilities. This was not efficient and resulted in slow scripts

for normal machines with a limited amount of CPU power and therefore stalled the analysis. By using the combinatorial identities and simplifications seen in Section 4.1.3.1, the computation became a multiplication of a binomial CDF and a combinatorial division, which eliminated the double nested loop and made the analysis more efficient.

6.2.2 Finding set size S

As previously discussed, finding an optimal combination of p, Q, and S is non-trivial. It can be implemented using the same approach as for finding optimal values of p and Q, with the additional requirement that there exists an S satisfying the condition in Equation 4.3 or Equation 4.4, depending on the security requirement. This introduces additional CDF evaluations and binomial ratio computations that must be performed for every possible value of S for each valid pair (p, Q), increasing the time complexity by $O(pN - Q)$ for every valid (p, Q) pair.

In contrast, evaluating the model for fixed values of p and Q is considerably simpler, with a time complexity of $O(pN - Q)$ when searching over all possible values of S. Furthermore, an implementation that fixes p, searches for the largest S satisfying the requirement, and then evaluates every Q from 0 to S has overall time complexity $O(pN)$.

6.2.3 Choice of Programming Languages

Since previous work in this field, such as Tail-Hammer, was implemented in Python, we also chose Python for the scripts used to find values of p, Q, and S that satisfy both security and liveness guarantees. This was a natural choice because the `gmpy2` library provides support for high-precision numerical computations.

After completing the Python implementation, we wanted to compare the results with those obtained using another programming language in order to cross-validate the computations. A natural alternative was R, since it is widely used in statistics and therefore well suited for the probabilistic analysis in this work. In addition, R provides the high-precision arithmetic library `Rmpfr`, an extension of `gmp`, which was beneficial for our implementation.

An advantage of having implementations in both languages is that future researchers can continue this work using their preferred programming environment, or use both implementations for verification purposes. Due to time constraints, however, only one setting for finding optimal p and Q was implemented in R: the *at least one honest participant* case in a reliable broadcast network. This was also one of the faster settings to evaluate, which made it suitable for testing during the later stages of the project.

6.2.3.1 Implementation of binomial CDF

Neither `gmpy2` nor `Rmpfr` have a high-precision implementation of the cumulative distribution function. A precise and fast CDF implementation is needed in all models, however, it is especially used in the security equation 4.3 with S. There, it was only implemented in Python using `gmpy2`, with a recursive evaluation of the binomial PMF. This is a numerically stable way of computing the CDF. However, one potential issue could be that each iteration introduces a tiny rounding error, which over many steps can accumulate. The complexity also grows linearly with the number of iterations.

In our R implementation, the `pbetaI` function is used. It evaluates the same quantity through the regularised incomplete beta function, as described at the end of Section 3.3.1, using specialised numerical methods with higher performance. So the security condition with S might benefit from being implemented in R with `Rmpfr` for larger runs, like trying to find optimal p, Q, and S.

6.3 Future work

In this section, we discuss some possible extensions of the work. We first look at how S, p, and Q could be jointly optimised, and then consider how inactive participants could be included in the analysis.

6.3.1 Optimising S, p, and Q

As mentioned previously, this work does not include finding an optimal S, p, and Q jointly. As discussed, different trade-offs are shown in Figures 5.1-5.7. The optimal selection of variables with a set size S included can be approached in different ways. One could focus solely on finding S and p to minimise Q, which would reduce the computational cost for each participant. However, as Figure 5.7 suggests, when S is chosen as the largest set size satisfying the liveness requirement, Q decreases rapidly as p grows. A larger S and p, however, would increase communication costs. Future work in this direction would need to examine these trade-offs and develop a computationally feasible method for jointly determining these three optimal variables without excessive time complexity.

6.3.2 Including Inactive Participants

The hypergeometric quorum model could be extended to introduce the inactive participants discussed in Section 3.1.5, which would split the total population of participants into honest H, malicious M, and inactive I with the total population being $N = H + M + I$. This would lead to the committee selection following a multivariate hypergeometric distribution instead, described in Section 3.3.3.

We investigated this extension and found that, under our current assumptions, the multivariate model reduces to the former hypergeometric case. The reason is that inactive participants, by definition, do not participate in the committee. They neither vote nor verify quorums. Their presence in the population is therefore the same as reducing the total population size from N to $N - I$, while the number of malicious participants M stays the same. After applying combinatorial identities, the resulting expression simplifies to the same hypergeometric distribution already used in our analysis, with parameters $(N - I, M, Q)$.

This means that under the assumption that inactivity is independent of whether a participant is honest or malicious, inactive participants affect the protocol only by reducing the pool of honest participants available to form quorums. The mathematical framework does not need to change, only the effective value of H decreases.

However, if inactive participants have other qualities, such as different types of malicious inactive participants, the multivariate hypergeometric distribution could become relevant, though this lies outside the scope of this work.

6.3.3 Extending the R implementation

As discussed in Section 6.2.3, only the *at least one honest* reliable broadcast case was reimplemented in R, due to time constraints. Extending the R implementation to cover the remaining settings (asynchronous network, *honest majority*, and the S-related security conditions) would provide independent validation of all numerical results in this work, not just the reliable broadcast *at least one honest* case. The R implementation could also offer practical performance benefits for the more computationally expensive cases. The `Rmpfr` package provides a high-precision implementation of the regularised incomplete beta function through `pbetaI`, which evaluates the binomial CDF in a single call rather than the iterative term by term recurrence used in the Python implementation. This is particularly relevant for the joint optimisation of p, Q, and S discussed in Section 6.3.1, where the additional nested search over S makes the Python implementation slow.

7
Conclusion

Distributed systems that rely on consistency for security, including key transparency and certificate transparency systems, face an important challenge in balancing security and efficiency. Protocols using anonymous committees, where a randomly selected subset of participants votes for each state, distribute trust without relying on a central authority that would otherwise be a natural target for adversaries. The efficiency of such protocols depends on two parameters: the selection probability p and the quorum threshold Q. Smaller values of both reduce computational and communication costs, but choosing them too small risks breaking either security or liveness. This thesis investigated how to set these parameters more precisely than prior work, and how to extend the analysis when participants depend on a potentially malicious central server to forward votes.

The first research question asked whether a smaller quorum size Q could be achieved by modelling the number of malicious participants in a quorum more precisely than the binomial approximation used in prior work, while maintaining the same security guarantees. This was addressed by modelling the number of malicious participants in a quorum drawn from a committee as a hypergeometric distribution, which exactly models sampling the quorum without replacement from the finite committee. The hypergeometric model produces tighter probability bounds, which allow both a smaller expected committee size and a smaller quorum. For the *at least one honest* reliable broadcast setting with 128-bit security and a population of $N = 10^9$, the quorum is reduced from $Q = 481$ in Tail-Hammer to $Q = 74$ in this work, and the expected committee size $C = pN$ from 828 to 249, reductions of approximately 85% and 70% respectively. The smaller Q means fewer cryptographic verifications per participant per epoch, and the smaller C means fewer votes in total, lowering the overall bandwidth usage.

The second research question asked how the minimum number of votes S that a participant must receive from a potentially malicious central server should be set to maintain security and liveness. By requiring at least S votes before accepting a state, a participant limits the Channel Maintainer's influence, preventing it from reducing the effective committee size below what is needed to reach a secure quorum of size Q.Security and liveness conditions on S were derived assuming the worst case where the Channel Maintainer delivers all malicious votes together with a subset of the honest ones. The conditions can be satisfied individually for a given p and Q, but the analysis revealed that parameters optimised independently are not jointly compatible with any feasible S. Fixing p first and deriving S and Q jointly shows

a clear trade-off: as p and S grow, the smallest valid Q shrinks, revealing a tension between bandwidth, which scales with p and S, and per-participant verification cost, which scales with Q. A joint optimisation over all three parameters would make this trade-off explicit but is computationally expensive and remains as future work.

The third research question asked whether the parameters p and Q obtained from the Python implementation could be validated independently in R. This was addressed by re-implementing the *at least one honest* reliable broadcast setting using the `Rmpfr` package, which provides arbitrary-precision arithmetic via the same `MPFR` library that underlies the `gmpy2` library used in Python. The R implementation reproduced the same optimal values of p and Q as the Python script across all tested population sizes (Table 5.2), providing independent confirmation of the numerical results. Beyond cross-validation, the R implementation also points to a practical improvement of future work. The `Rmpfr` package provides a high-precision implementation of the regularised incomplete beta function through `pbetaI`, which evaluates the binomial CDF in a single call rather than the iterative term by term recurrence used in the Python implementation. Implementing the more computationally expensive cases (in particular those involving S) in R could therefore make the joint optimisation of p, Q, and S identified above easier.

A practical limitation encountered during this work was that the high-precision libraries used for evaluating failure probabilities (`gmpy2` in Python and `Rmpfr` in R) run on the CPU and cannot take advantage of GPU acceleration. This made parallelisation difficult on standard machines and contributed to long runtimes, particularly for the *honest majority* cases. In a deployed protocol this cost is paid only once, since the resulting p and Q depend only on the population size, the malicious fraction, and the security parameter, and could be precomputed and looked up from a table. All results in this thesis are theoretical, and an empirical evaluation in a deployed protocol such as CoD would be needed to confirm the predicted reductions in computational and communication overhead.

Across all three research questions, the main idea was to model the quorum's composition as drawn from the committee that actually formed, rather than approximating it as an independent sample from the full population. In prior work, the number of malicious participants in the quorum is modelled as a binomial random variable parametrised by the population, which treats committee selection and quorum sampling as if they were the same step. By instead conditioning on the committee and using a hypergeometric distribution of the quorum, the analysis captures that the quorum is drawn without replacement from a finite, already-realised set of committee members. This yields tighter probability bounds for the same security level, and in turn, the reductions in Q and C reported above. The same conditioning approach extends naturally to the asynchronous setting and to the set size S introduced for an unreliable Channel Maintainer, and provides a foundation on which future work can build a joint optimisation for all three parameters.

Bibliography

[1] Algorand Foundation. Algorand Specifications, 2025.

[2] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 10 1985.

[3] Apple. Advancing iMessage security: iMessage Contact Key Verification - Apple Security Research, 10 2023.

[4] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a Public Blockchain Keep a Secret? In *Theory of Cryptography (2020), Lecture Notes in Computer Science*, volume 12550, pages 260–290. Springer, Cham, 12 2020.

[5] Erica Blum, Derek Leung, Julian Loss, Jonathan Katz, and Tal Rabin. Analyzing the Real-World Security of the Algorand Blockchain. In *CCS 2023 - Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 830–844. Association for Computing Machinery, Inc, 11 2023.

[6] Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 11 1987.

[7] Nicholas Brandt, Mia Filić, and Sam A Markelon. SoK: On the Security Goals of Key Transparency Systems. *Cryptology ePrint Archive*, (Paper 2024/1938), 2024.

[8] Joakim Brorsson, Elena Pagnin, Bernardo David, and Paul Stankovski Wagner. Consistency-or-Die: Consistency for Key Transparency. *Cryptology ePrint Archive*, (Paper 2024/879), 2024.

[9] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10821 LNCS:66–98, 2018.

[10] Bernardo David, Lucia Lavagnino, Elena Pagnin, and Paul Stankovski Wagner. Tail-Hammer: Optimized statistics for anonymous committees and applications. In *Proceedings of the 15th International Conference on Security and Cryptography for Networks (SCN 2026)*. Springer, 2026. To appear.

[11] Bernardo David, Bernardo Magri, Christian Matt, Jesper Buus, Nielsen ‡4, and Daniel Tschudi. GearBox: Optimal-size Shard Committees by Leveraging the Safety-Liveness Dichotomy. *Cryptology ePrint Archive*, 2021.

[12] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. Cryptology ePrint Archive, Paper 2004/310, 2004.

[13] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 4 1985.

[14] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2), 6 2007.

[15] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You Only Speak Once: Secure MPC with Stateless Ephemeral Roles. *Lecture Notes in Computer Science*, 12826 LNCS:64–93, 8 2021.

[16] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. *Cryptology ePrint Archive*, 2017.

[17] Oded. Goldreich. *Foundations of cryptography.* Cambridge University Press, Cambridge, UK, 2006.

[18] N L Johnson, S Kotz, and A W Kemp. Univariate discrete distributions. page 565, 1992.

[19] Jonathan. Katz and Yehuda. Lindell. *Introduction to modern cryptography.* CRC Press, 3rd edition, 2021.

[20] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10401 LNCS:357–388, 2017.

[21] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 7 1982.

[22] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. (RFC 6962), 6 2013.

[23] Ben Laurie. Certificate transparency. *Communications of the ACM*, 57(10):40–46, 9 2014.

[24] Martin Maechler. Interface R to MPFR - Multiple Precision Floating-Point Reliable [R package Rmpfr version 1.1-2]. *CRAN: Contributed Packages*, 10 2025.

[25] Dahlia Malkhi and Michael Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[26] Alex Martelli and Case Van Horsen. gmpy2, 2026.

[27] Melara M. S., A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman. *CONIKS: Bringing Key Transparency to End Users*. USENIX Association, 8 2015.

[28] Meta. WhatsApp Key Transparency Overview. 8 2023.

[29] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. *Annual Symposium on Foundations of Computer Science - Proceedings*, pages 120–130, 1999.

[30] Frank W. J. Olver, Adri B. Olde Daalhuis, Daniel W. Lozier, Barry I. Schneider, Ronald F. Boisvert, Charles W. Clark, Bruce R. Miller, Bonita V. Saunders, Howard S. Cohl, and Marjorie A. McClain. NIST Digital Library of Mathematical Functions, §8.17 Incomplete Beta Function, 3 2026.

[31] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM (JACM)*, 27(2):228–234, 4 1980.

[32] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure messaging. *Proceedings - IEEE Symposium on Security and Privacy*, 2015-July:232–249, 7 2015.

[33] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, Inc., 3rd edition, 1968.

A

Python and R Implementation

This chapter contains the Python and R functions referenced in Section 4.2.

Listing A.1: `binom_cdf_leq`

```
def binom_cdf_leq(q, n, prob):
    q = int(q)
    n = int(n)
    prob = mpfr(prob)

    if q < 0:
        return mpfr(0)
    if q >= n:
        return mpfr(1)
    if prob == 0:
        return mpfr(1)
    if prob == 1:
        return mpfr(0) if q < n else mpfr(1)

    one_minus_p = mpfr(1) - prob
    ratio = prob / one_minus_p

    pk = one_minus_p ** n
    cdf = pk

    for k in range(q):
        pk *= mpfr(n - k) / mpfr(k + 1) * ratio
        cdf += pk

    return cdf
```

Listing A.2: `binom_cdf_geq`

```
    def binom_cdf_geq(Q, N, p):
    Q = int(Q)
    N = int(N)
    p = gmpy2.mpfr(p)

    if Q <= 0:
        return ONE
    if Q > N:
        return ZERO
```

```
10
11      return ONE - binom_cdf_leq(Q - 1, N, p)
```

Listing A.3: comb_div

```
1       def comb_div(M, N, Q):
2       M = int(M)
3       N = int(N)
4       Q = int(Q)
5
6       if Q < 0 or Q > M or Q > N:
7           return ZERO
8       if Q == 0:
9           return ONE
10
11      r = ONE
12      for t in range(Q):
13          r *= gmpy2.mpfr(M - t) / gmpy2.mpfr(N - t)
14      return r
```

Listing A.4: prob_X_eq_Q

```
1       def prob_X_eq_Q(N, M, p, Q):
2       N = int(N)
3       M = int(M)
4       Q = int(Q)
5       p = gmpy2.mpfr(p)
6
7
8       if Q < 0 or Q > M or Q > N:
9           return ZERO
10
11      return binom_cdf_geq(Q, N, p) * comb_div(M, N, Q)
```

Listing A.5: hypergeom_tail_population_gmp

```
1       def hypergeom_tail_population_gmp(N, K, Q, q):
2       N = int(N)
3       K = int(K)
4       Q = int(Q)
5       q = int(q)
6       G = N - K
7
8       if q <= 0:
9           return ONE
10      if q > Q or Q > N:
11          return ZERO
12
13      den = gmpy2.bincoef(N, Q)
14      total = ZERO
15
```

```
    for x in range(q, Q + 1):
        if x > K:
            continue
        if Q - x > G:
            continue
        num = gmpy2.bincoef(K, x) * gmpy2.bincoef(G, Q - x)
        total += gmpy2.mpfr(num) / gmpy2.mpfr(den)

    return total
```

Listing A.6: `find_best_n`

```
    def find_best_n(N, fM, n_grid, b, base_window=64):
    total = len(n_grid)

    for idx, n in enumerate(n_grid):
        if n > N:
            continue

        p = gmpy2.mpfr(n) / gmpy2.mpfr(N)

        if (idx + 1) % 50 == 0 or idx == 0:
            print(f" [{idx+1}/{total}] n={n}, p~{float(p):.6e} ...", flush=
                True)

        res = find_Q_minimax_bound(N, fM, n, b, p, base_window=base_window)

        if res["Q"] is not None:
            log2_pf = gmpy2.log2(res["p_fail"]) if res["p_fail"] > 0 else
                None
            return {
                "N": N,
                "fM": fM,
                "p": p,
                "n": n,
                "b": b,
                "bound": res["bound"],
                "Q": res["Q"],
                "p_honest_majority_fail": res["p_honest_majority_fail"],
                "p_async_split_fail": res["p_async_split_fail"],
                "p_liveness_fail": res["p_liveness_fail"],
                "p_fail": res["p_fail"],
                "log2_p_fail": log2_pf,
            }

    return None
```

Listing A.7: `binom_ratio`

```
    def binom_ratio(M, Q, S):
    r = mpfr(1)
```

```
    for t in range(Q):
        r *= mpfr(M - t) / mpfr(S - t)
    return r
```

Listing A.8: binom_gmp

```
def binom_gmp(n, k):
n = int(n)
k = int(k)
if k < 0 or k > n or n < 0:
    return mpfr(0)
return mpfr(gmpy2.comb(n, k))
```

Listing A.9: security_sum

```
def security_sum(M, Q, S, p):
M, Q, S = int(M), int(Q), int(S)
b_min = math.floor(Q / 2)
denom = binom_gmp(S, Q)

one_minus_p = mpfr(1) - p
ratio = p / one_minus_p

pk = one_minus_p ** M
pmf = [pk]
for j in range(1, S + 1):
    pk = pk * mpfr(M - j + 1) / mpfr(j) * ratio
    pmf.append(pk)

prob_tail = mpfr(1) - sum(pmf)
if prob_tail < 0:
    prob_tail = mpfr(0)

if denom == 0:
    return prob_tail

total = mpfr(0)
for i in range(b_min, Q + 1):
    j_lo = i
    j_hi = min(S, S - Q + i)
    if j_hi < j_lo:
        continue
    for j in range(j_lo, j_hi + 1):
        total += binom_gmp(j, i) * binom_gmp(S - j, Q - i) * pmf[j]

return total / denom + prob_tail
```

Listing A.10: find_S

```
def find_S(N, M, p, threshold):
for S in range(int(N * p), 0, -1):
```

```
3         if binom_cdf_leq_gt(S, M, p) < threshold and binom_cdf_leq(S-1, N,
              p) < threshold:
4             return S
5     return None
```

Listing A.11: `max_prob`

```
1     def max_prob(M, Q, S, p):
2     p_pow_Q = p ** Q
3     M_async= M + (N-M)/2
4     prob = max(security_sum(M, Q, S, p), binom_ratio(M_async, Q, S) *
          p_pow_Q * binom_cdf_leq(S-Q, M_async-Q, p) + binom_cdf_leq_gt(S,
          M_async, p))
5     return prob
```

Listing A.12: `binom_cdf_leq` (R script)

```
1  binom_cdf_leq <- function(q, n, prob) {
2    q <- as.integer(q)
3    n <- as.integer(n)
4
5    if (q < 0L) return(ZERO)
6    if (q >= n) return(ONE)
7
8    p <- mpfr(prob, precBits = PREC)
9    if (p == 0) return(ONE)
10   if (p == 1) return(if (q < n) ZERO else ONE)
11
12   pbetaI(ONE - p, mpfr(n - q, precBits = PREC), mpfr(q + 1L, precBits =
         PREC))
13 }
```

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY