Yale University
Department of Computer Science

Exposing Computationally-Challenged Byzantine Impostors

James Aspnes[1] Collin Jackson[2] Arvind Krishnamurthy[3]

Department of Computer Science
Yale University

YALEU/DCS/TR-1332
July 26th, 2005

[1]Email: aspnes@cs.yale.edu. Supported in part by NSF grants CCR-0098078 and CCR-0305258.

[2]Email: collin.jackson@gmail.com.

[3]Email: arvind@cs.washington.edu. Supported in part by NSF grants CCR-9985304, ANI-0207399, and CCR-0209122.

Exposing Computationally-Challenged Byzantine Impostors

James Aspnes* Collin Jackson† Arvind Krishnamurthy‡

Department of Computer Science
Yale University

Abstract

Internet protocols permit a single machine to masquerade as many, allowing an adversary to appear to control more nodes than it actually does. The possibility of such *Sybil attacks* has been taken to mean that distributed algorithms that tolerate only a fixed fraction of faulty nodes are not useful in peer-to-peer systems unless identities can be verified externally. The present work argues against this assumption, by presenting practical algorithms for the distributed computing problem of *Byzantine agreement* that defend against Sybil attacks by using moderately hard puzzles as a pricing scheme for identities. Though our algorithms do not prevent Sybil attacks entirely, they solve Byzantine agreement (and some useful variants) when the limited fraction of nodes that can fail is replaced by a limited fraction of the total computational power. These results suggest that Byzantine agreement and similar tools from the distributed computing literature are likely to help solve the problem of adversarial behavior by components of peer-to-peer systems.

1 Introduction

Peer-to-peer systems that allow arbitrary machines to connect to them are known to be vulnerable to *pseudospoofing* or *Sybil attacks*, first described in a paper by Douceur [7], in which Byzantine nodes adopt multiple identities to break fault-tolerant distributed algorithms that require that the adversary control no more than a fixed fraction of the nodes. Douceur argues in particular that no practical system can prevent such attacks, even using techniques such as pricing via processing [9], without either using external validation (e.g., by relying on the scarceness of DNS domain names or Social Security numbers), or by making assumptions about the system that are unlikely to hold in practice. While he describes the possibility of using a system similar to *Hashcash* [3] for validating identities under certain very strong cryptographic assumptions, he suggests that this approach can only work if (a) all the nodes in the system have nearly identical resource constraints; (b) all identities are validated simultaneously by all participants; and (c) for "indirect validations," in which an identity is validated by being vouched for by some number of other validated identities, the number of such witnesses must exceed the maximum number of bad identities. This result has been abbreviated by many subsequent researchers [8, 11, 19–21] as a blanket statement that preventing Sybil attacks without external validation is impossible.

We argue that this impossibility result is much more narrow than it appears, because it gives the attacking nodes a significant advantage in that it restricts legitimate nodes to one identity each. By removing this restriction we can resist the Sybil attack for the central problem of Byzantine agreement [13], in which all

*Email: `aspnes@cs.yale.edu`. Supported in part by NSF grants CCR-0098078 and CCR-0305258.

†Email: `collin.jackson@gmail.com`.

‡Email: `arvind@cs.washington.edu`. Supported in part by NSF grants CCR-9985304, ANI-0207399, and CCR-0209122.

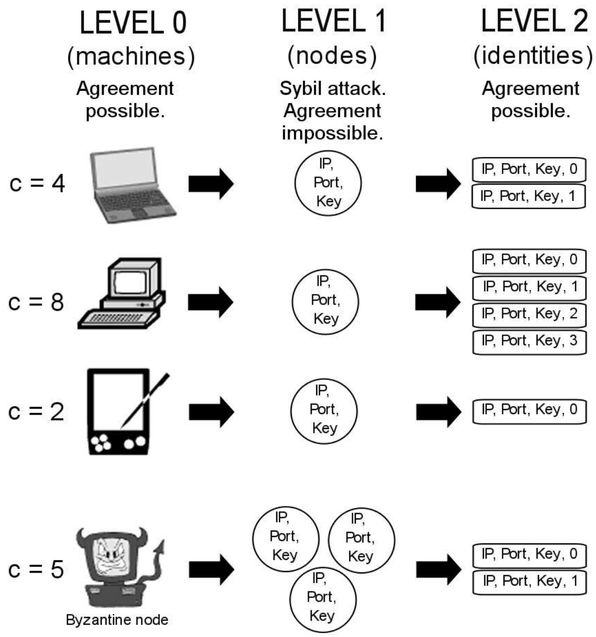

Figure 1: Byzantine agreement is not possible amongst nodes, but becomes feasible with priced identities.

non-faulty participants must agree on some single decision value despite the interference of faulty nodes. Though Byzantine agreement can be solved trivially in the model used in [7] (because that model provides synchronous reliable broadcast) we show that even in a standard synchronous message-passing model (without reliable broadcast) it can still be solved if we use digital signatures to enforce distinguishability between alleged identities. Many traditional distributed computing problems are solvable with Byzantine agreement protocols, so our algorithms can be used to accomplish a wide variety of objectives.

Our two algorithms in in Section 3 and Section 4 use *moderately hard puzzles* [3,12,18] as a demonstration of computing power. They are designed a preamble for any standard Byzantine agreement algorithm, and they create a virtual network where identities are priced by computing power so that consensus algorithms can safely run. This technique solves Byzantine agreement if the adversary controls less than a third of the total computational power in the system, and in the specific case where all machines have equal computational power, it achieves consensus with multiple identities per node under exactly the same conditions as it is solvable with single identities. It follows that for any problem that can be reduced to Byzantine agreement, our ability to solve that problem is not affected by allowing Byzantine nodes to masquerade as multiple nodes.

Note that standard Byzantine agreement places few constraints on the common decision value. In particular, the adversary can determine which value is decided on. For peer-to-peer applications, it is more natural to demand *strong consensus* [15], where the decision value must be the input of some good node, or δ-*differential consensus* [10], where the decision value must be nearly a plurality value among the good nodes. The virtual network created by our algorithms can be used as a preamble for strong and δ-differential consensus algorithms as well.

2 Model

We assume a synchronous point-to-point network with reliable messages, where machines have some source of nondeterminism for the generation of random numbers. Each machine may have multiple addresses, and there is no mechanism for distinguishing multiple machines with one address each from a single machine with many addresses. This assumption is justified in practice not only because IP addresses are easy to spoof, but because many machines now sit behind firewalls using Network Address Translation, which presents many machines on the inside of the firewall as a single machine to the outside.

We can imagine representing a *node* as an IP address and a port number, with the assumption (necessary to build any protocol at all) that the adversary cannot corrupt messages or arrange for messages directed at a particular address to be delivered elsewhere. To prevent spoofing of outgoing messages, we further imagine that each node chooses a public key that it appends to each outgoing message from that address, along with a digital signature for the message using the corresponding private key. We do not assume the presence of a public key infrastructure to guarantee that these public keys are not themselves spoofed, and in general a node can generate as many public keys as it wants; but recipients can treat messages arriving with different public keys as coming from different nodes, so the problem of pruning out extraneous public keys reduces to the problem of pruning out extraneous nodes.

We assume that each node has some limited amount of computing power, defined as the number of puzzle solutions that the node can generate in a single round, for any of the puzzles that are defined in Sections 3 and 4. There are N physical nodes in the network, N_g of which are good (non-faulty) and N_e of which are evil (Byzantine, i.e., controlled by the adversary). Let C be the total computing power of all the nodes in the physical network. The computing power C_g of the good nodes is fixed but not necessarily uniform. The collective computing power C_e of the adversary can be dynamically allocated among the adversary nodes. Our goal is to devise a pricing scheme for assigning *identities* to nodes, with the property that the proportion of identities belonging to good nodes at the end of the protocol is close to their share of the total computational power (as illustrated by Figure 1).

A final assumption is that the set of nodes participating in the protocol is known to each node at the start of the validation protocol, which means that we can order the nodes and assign them an index based on their IP address, port number, and digital signature. This assumption is necessary to allow the nodes to communicate at all given only a point-to-point message-passing network, but it does raise the question of how this agreed-upon set of nodes is determined and distributed to the nodes. We do not address this question at present, assuming simply that some centralized sign-up mechanism exists, but note that it does provide interesting possibilities for future work.

3 Democracy

The *Democracy* algorithm takes three rounds to validate identities. In the first round, each node sends an individualized sub-puzzle to every other node. In the second round, each node determines its puzzle from the sub-puzzles, computes as many solutions as possible, and sends the puzzle and its solutions back to every node in the system. In the final round, each node verifies the received solutions and assigns the correct number of identities to that node, handing control of subsequent protocol interactions over to its own identities.

Since the adversary can only help itself by sending correct solutions when such a solution is available, we can safely assume that it sends each of its solutions to every good node. Let ϵ be the expected amount of computational power required to acquire an identity. If the sub-puzzles cannot be cheated as discussed below, the expected number of identities assigned to adversarial nodes, I_e, is C_e/ϵ and the expected number of identities assigned to legitimate nodes, I_g, is C_g/ϵ. If $C_e < C_g/2$, then $I_e < I_g/2$ (allowing unauthenticated consensus), and if $C_e < C_g$, then $I_e < I_g$ (allowing δ-differential consensus).

Democracy only works if moderately hard puzzles can be constructed from a number of sub-puzzles, many of which are chosen by the adversarial nodes. We present a puzzle approach that provides the desired guarantees:

- **Parameters**: A one-way hash function H. Its domain is bit-strings of length $N\,S + k$ and the range includes strings of lengths greater than w, where $S \geq k > w > 0$.
- **Input**: The puzzle string y is of length NS and contains the S bits received from each node, ordered by node index.
- **Puzzle**: Compute as many x_i as possible such that the most significant bits of each $H(y \mid x_i)$ are 0^w.
- **Output**: Send y and all the x_i to every node.
- **Verification**: Check the appropriate portion of y for the sent bits. Check that the most significant bits of each $H(y \mid x_i)$ are 0^w.

Using this puzzle scheme, the *Democracy* algorithm sends $O(N_g\,N)$ messages, ignoring messages between adversarial nodes. Sending the string y requires $O(N\,S)$ bits, and sending the solutions requires $O(k\,C_{\max}) = O(S\,C_{\max})$ bits, where $C_{\max}$ is the number of puzzles the most powerful node in the system can solve.

The hash function should be a cryptographically strong function such as MD5 or SHA1. We assume that no attack on the hash function can produce a puzzle solution faster than trying random inputs, even when the adversary can fix some of the input bits. Though this is a common assumption in the literature [2, 4, 5, 16], it should be noted that it is a very strong property, and while it holds for *random functions*, it is not known whether standard cryptographic hash functions provide such security.[1] The value of w should be chosen to be small enough so that a node with computing power ϵ can compute one x_i on average during the time allotted for puzzle solving. Since the expected number of solutions (and hence, the verification time) is proportional to 2^{-w}, w should be large enough that every node will have time to verify every identity during the verification round.

We note that this way of combining the sub-puzzles into one puzzle for *Democracy* has the following desirable properties:

- It is resistant to tampering. The adversary cannot discredit other legitimate nodes by supplying impossible or confusing sub-puzzles. Any string of bits is a valid sub-puzzle, and if the correct number of bits is not received, 0 can be used as a placeholder.
- It is resistant to precomputation. An adversary would need to create a table of size 2^S to store the solutions for every string of input bits. Furthermore, this table would only succeed in fooling one node, so a table of size $N_g\,2^S$ would be needed to convince all good nodes.[2]
- It is resistant to collusion. Although the adversary can choose many of the sub-puzzles, it does not control all of them. If the output of MD5 or SHA1 is computationally indistinguishable from random, the adversary's ability to control some of the input bits will not make finding collisions easier.

[1] This fact was observed by Douceur [7], who proposed a similar puzzle problem without constraining the order of combining different identities' contributions to the puzzle. Douceur observed that *partial-preimage resistance* was a minimum requirement for such a puzzle, but because we control only part of the output the full requirements are even stronger. See Menezes et al. [14] Remark 9.22 and Section 9.5.2 for a definition of partial-preimage resistance and a discussion of the difficulties of applying cryptographic hash functions in applications of this sort.

[2] To make the precomputation table size $2^{N_g S}$, one might require nodes to send the digitally signed versions of their input bits. However, this approach makes verification more expensive and it destroys the tamper-resistant property, because Byzantine nodes can discredit good nodes by not sending any sub-puzzles. The bits spent providing the digitally signed version of S would be better spent making it larger.

- It is scalable. If one computer can find c collisions in one round on average, we would expect two such computers running side-by-side and searching different parts of the function space to find $2c$ collisions on average. Furthermore, the time to compute MD5 or SHA1 fingerprints does not depend significantly on the input bits, even if the puzzles are different, so we could hand the computers different puzzles and the expected number of collisions would still be twice as many.

4 Monarchy

The *Democracy* algorithm in Section 3 prices identities in a constant number of rounds, but at the cost of making strong assumptions about the underlying puzzle. We provided a puzzle approach, but we note that the running time to compute solutions for the posed puzzles are modeled by a probabilistic cost function. In this section, we propose a different algorithm that provides more flexibility in what puzzles can be used, thereby allowing us to employ puzzles that have fixed running costs, such as the *time-lock puzzle* of [18].

In the *Monarchy* algorithm, each node takes its turn sending puzzles and receiving solutions. If r is the round number, the "king" of the round is the node with index r. The king sends out an individualized puzzle to each node at the end of the round before he is king. (We include a round -1 so that the king with index 0 can send his puzzles.) Each node finds as many solutions to the puzzle as possible during the king's round and sends the solutions to the king at the end of that round.

This process continues up until round N, when the nodes stop solving problems and spend the round verifying the solutions sent to them. If node n_β sends c_β solutions to n_α, then n_α assigns c_β/ϵ identities to n_β in the virtual network. Each node broadcasts how many identities it thinks each other node has, and then hands over control of subsequent protocol interactions to its identities.

Table 1: A comparison of *Monarchy* and *Democracy*

	Rounds	Messages	Message Size	Fault Tolerance (unauthenticated)	Fault Tolerance (δ-consensus)
Monarchy	$O(N)$	$O(N_g N)$	$O(S\,C_{\text{max}} + C\log(N)/\epsilon)$	$C_g/3$	$C_g/2$
Democracy	$O(1)$	$O(N_g N)$	$O(S\,C_{\text{max}} + NS)$	$C_g/2$	C_g

We treat the identities as autonomous agents hosted by the nodes. Identity $n_{\alpha,i}$, the i^{th} identity hosted by node n_α, inherits its initial notions of trust from its host node: $n_{\alpha,i}$ begins by trusting $n_{\beta,j}$ if n_α assigns at least j identities to n_β. It then interprets the identity assignments broadcast by other host nodes as "accusations." If $n_{\gamma,k}$ is trusted by $n_{\alpha,i}$ and n_γ attributes less than j identities to n_β, then $n_{\gamma,k}$ is accusing $n_{\beta,j}$ to be illegitimate. If $n_{\alpha,i}$ receives more than C_e/ϵ such accusations regarding $n_{\beta,j}$'s legitimacy, then $n_{\alpha,i}$ stops trusting $n_{\beta,j}$. Assuming that the legitimate identities begin by trusting other legitimate identities, the adversaries cannot confuse a legitimate identity $n_{\alpha,i}$ into losing trust in another legitimate identity $n_{\beta,j}$ unless it can convince n_α to assign more than C_e/ϵ identities to the adversarial nodes. However, if the puzzles were well-designed, the adversarial nodes will not be able to find more than C_e solutions, and thus cannot obtain more than C_e/ϵ identities from n_α.

To remain in the virtual network, each adversarial identity must be trusted by at least $I_g - C_e/\epsilon$ legitimate identities, so that no more than C_e/ϵ legitimate identities will accuse it by not listing it as trusted. Since good nodes follow the protocol and earn their identities honestly, we expect $I_g = C_g/\epsilon$. Each of these C_g/ϵ legitimate identities can be fooled by at most C_e/ϵ adversarial identities. Hence, at most $(C_g/\epsilon)(C_e/\epsilon)/(C_e/\epsilon - C_g/\epsilon)$ adversarial identities can survive accusations. It follows that if $C_e < C_g/3$, then $I_e < I_g/2$ (allowing unauthenticated consensus), and if $C_e < C_g/2$, then $I_e < I_g$ (allowing δ-differential consensus).

It may appear as though increased fault tolerance can be achieved with further rounds of accusations, but this is not the case. The adversary can only hurt itself by accusing other adversarial nodes, so we can safely assume the adversary never accuses itself. Furthermore, good nodes pay for their identities through honest work, so good identities will generally not accuse other good identities. If enough good identities accuse an enemy identity $n_{\beta,j}$ to convince any one good identity that previously trusted $n_{\beta,j}$, then *all* good identities will be similarly convinced. Otherwise, the same number of accusations against $n_{\beta,j}$ exist after the accusation round, so further rounds of accusations would be uneventful.

There are $O(N_g\, N)$ messages total, ignoring messages between adversarial nodes. If the problems and solutions each require $O(S)$ bits to communicate, where S is a security parameter, the message size during each king's round will be $O(SC_{\text{max}})$, where C_{max} is the computational power of the most powerful node. The messages in the final round require $\min(N \log(I/N), I \log(N))$ bits on average as a host node can either send the number of identities assigned to each node or it can send the node associated with each identity. We note that the adversary has control over N whereas no legitimate node would claim I exceeds C/ϵ, so the system can predictably bound message size to $C \log(N)/\epsilon$ by choosing to communicate the node associated with each identity.

5 Related work

The technique of using moderately hard problems to limit an adversary's abuse of resources was first suggested by Dwork and Naor [9] as a method for combating junk email ("spam"). However, the distributed consensus problem differs from junk email problem in that legitimate participants must prove their identity to adversaries as well as other legitimate participants. A legitimate node that is overwhelmed demonstrating its computational power to everyone at once may lose the trust of its legitimate peers, so some amount of coordination is required to ensure that no unreasonable demands are made of legitimate nodes.

Using moderately hard problems as a defense against denial-of-service attacks has been suggested by Juels and Brainard [12] and by Back [3]. Increasing the difficulty of the puzzles in an attack situation allows for graceful degradation of server performance. Back's *Hashcash*, on which we base our puzzle scheme, is designed as a challenge-response protocol between two parties, but adapts well to a distributed setting.

Rivest, Shamir, and Wagner [18] present a time-lock puzzle that works well when there are only two parties involved, but in contrast to *Hashcash*, it is difficult to decompose into sub-puzzles because verification requires secret information about the factorization of a product of primes. Dwork and Naor's suggestion of square roots modulo a prime [9] also looks promising, but is not necessarily collusion-proof; an adversary that can choose which number to take the square root of in some prime field may be able to determine which one will be easier to compute.

In heterogeneous environments, there are often vast differences in computational power between devices. High-performance workstations with specialized hardware can solve many more problems than PDAs, for example. Abadi *et al.* recently presented a moderately hard puzzle class with solution times that depend on memory access times rather than clock speed [1]. Their approach adapts well to our algorithm and would be an effective defense against malicious, high-end attacks.

6 Conclusion

We have described two algorithms for limiting the effect of multiple identities in a peer-to-peer system. These algorithms have complementary strengths; the *Democracy* algorithm of Section 3 is faster and tolerates more faults than the *Monarchy* algorithm of Section 4, but at the cost of larger messages (as summarized by Table 1) and stronger requirements for the embedded puzzle problem.

By using the Democracy or Monarchy algorithms as a preamble, we can solve Byzantine agreement despite the efforts of Byzantine nodes with multiple identities. Our algorithms are also relevant in a number of settings such as self-policing peer-to-peer systems that detect Byzantine agents and freeloaders [19, 20], distributed trust management systems [8], toolkits for building high-integrity services [17], and Byzantine fault-tolerant distributed file systems [6].

An obvious question is whether some hybrid algorithm could combine the positive features of both algorithms. Other questions are whether the complexity could be further reduced with other cryptographic primitives, how to bootstrap the initial assumption that all claimed identities are known, what lower bounds can be proved to show the potential scope of this approach, and what practical issues arise if these techniques are implemented. We plan to address all of these questions in future work.

References

[1] Martín Abadi, Michael Burrows, and Ted Wobber. Moderately hard, memory-bound functions. In *NDSS*. The Internet Society, 2003.

[2] Rakesh Agrawal, Alexandre Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD international conference on on Management of data*, pages 86–97, 2003.

[3] Adam Back. Hashcash. Available at `http://www.cypherspace.org/hashcash/`, May 1997.

[4] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993.

[5] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. *Lecture Notes in Computer Science*, 1070, 1996.

[6] Castro and Liskov. Practical byzantine fault tolerance. In *Symposium on Operating Systems Design and Implementation*, 1999.

[7] John R. Douceur. The Sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, 2002.

[8] B. Dragovic, E. Kotsovinos, S. Hand, and P. R. Pietzuch. Xenotrust: Event-based distributed trust management. In *Proceedings of International Workshop on Database and Expert Systems Applications*, 2003.

[9] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology: CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer-Verlag, August 1992.

[10] Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In *Proceedings of the Twenty-Second ACM Symposium on the Principles of Distributed Computing*, pages 211–220, July 2003.

[11] Nicholas J. A. Harvey, John Dunagan, Michael B. Jones, and Stefan Saroiu. Skipnet: A scalable overlay network with practical locality properties. Technical Report MSR-TR-2002-92, Microsoft Research, 2002.

[12] A. Juels and J. Brainard. Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks. In *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, 1999.

[13] L. Lamport, R. Shostack, and M. Pease. The Byzantine generals problem. *ACM Transactions on Proggramming Languages and Systems*, 4(3):382–401, 1982.

[14] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[15] Gil Neiger. Distributed consensus revisited. *Information Processing Letters*, 49(4):195–201, 1994.

[16] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(3):361–396, 2000.

[17] Michael K. Reiter. The rampart toolkit for building high-integrity services. In *Theory and Practice in Distributed Systems*, volume 938. Springer-Verlag, Berlin Germany, 1995.

[18] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, 1996.

[19] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gun Sirer. Karma : A secure economic framework for peer-to-peer resource sharing. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[20] Marc Waldman. *Secure and Robust Censorship-Resistant Publishing Systems*. PhD thesis, New York University, May 2003.

[21] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. To appear, *IEEE Journal on Selected Areas in Communications*, 2003.