Survivability: Beyond Fault Tolerance and Cryptography

Lidong Zhou
Microsoft Research Silicon Valley
lidongz@microsoft.com

Survivability of a distributed system is the system's ability to function as expected despite adverse events such as component failures and malicious attacks. By definition, survivability demands reliability and security—two subjects that have been studied in isolation for a long time. On the one hand, the study on reliability fosters research on fault tolerant distributed systems, which centers around the notion of replication and consensus. Various system models have been proposed to capture different types of failures and timing assumptions. On the other hand, security requirements such as confidentiality, integrity, and authenticity have yielded cryptographic building blocks such as encryption/decryption schemes, one-way hash functions, and digital signatures.

A misconception naturally arises that a simple integration of fault tolerance mechanisms (such as replicated state machines [13, 15, 4] and Byzantine quorum systems [14]) and cryptography leads to a survivable distributed system. This is unfortunately not true. System models in fault tolerance traditionally rely on the probability distribution of random faults. Malicious attacks, by exploiting worse-case scenarios, often render such statistical analysis inappropriate, thereby debasing the foundation of existing fault tolerance solutions. Replication, a key element for fault tolerance, dictates a distributed architecture that requires protocols for multiple parties. Although such problems have been studied in cryptography as secure multiparty computation [7], the proposed schemes are not nearly as practical and accessible as those for encryption/decryption and digital signatures. Furthermore, survivability raises new types of concerns (such as denial-of-service attacks) that are beyond the current means of fault tolerance and cryptography. This position paper outlines some new challenges posed by survivability, as well as opportunities and new research directions.

1 Revisiting the model and the assumptions

System models are widely used for system designers to make simplifying assumptions about a system in order to focus on the most important aspects of the system, while ignore the unimportant ones. Such simplification is a powerful tool to manage complexity, but the resulting system models are often tightly coupled with a particular environment and objective. It is therefore crucial to revisit the system models and re-assess the assumptions when we attempt to reuse protocols in a different setting with different objectives.

Threshold failure model. Consider a system implemented by n replicas and call each replica a server. The widely used *threshold failure model* stipulates that the number of faulty servers does not exceed a threshold t. As pointed out previously [11], this model is flawed when we consider compromises due to malicious attacks, mainly because of correlations among server compromises. There are two main approaches to tackling this problem. One is to capture the correlations in a more general adversary model that specifies which subsets of servers might all be compromised, and retrofit the existing protocols for the general adversary model [10]. To come up with an appropriate adversary model, one could partition the servers along a set of attributes. For example, all servers running on the same hardware/software platform constitutes a set in the adversary model because one vulnerability in the platform takes down all these servers; so do all servers under the same administrative domains or within the same subnet.

An orthogonal approach aims at reducing or even eliminating correlations among the servers. Developing multiple implementations for diversity is often prohibitively expensive and has known limits [12]. A more promising method is to inject diversity through randomization [5]. It is yet to be shown how effective this method is in terms of eliminating common vulnerabilities; the impact of this approach on system performance and system reliability is largely unknown.

Recovery and adaptation. Another simplifying assumption made in fault tolerance is to ignore the recovery issue. This is justifiable because usually recovery simply involves detecting, isolating, and replacing faulty components. But the importance of recovery to reliability cannot be overstated: the purpose of fault tolerance is to offer a window of opportunity for recovery of faulty components. Analysis has shown that the recovery interval is a key factor in system reliability [19].

For survivability, which has to defend against malicious attacks, recovery becomes more challenging. First, compromise of components is hard to detect, so recovery only after detection of compromise might not suffice. Instead, recovery must be done periodically, in a proactive fashion. The lifetime of a system can then be partitioned into *generations* that are separated by periodic executions of recovery.

Furthermore, recovery of a compromised component goes beyond simply replacing the compromised components and restoring their states. Any secrets stored on a compromised component might have been exposed to the adversary and therefore must be made obsolete. In cases where the secrets are keys used for secure communication, the refreshing of secrets might involve re-establishing authenticated secure communication links [3] or refreshing secret shares of a key without changing the key [9].

Finally, if a component is compromised due to an adversary exploiting a vulnerability in that component, the same vulnerability remains after the recovery, opening the door for repeated compromises. Ideally, the component should evolve to be more resilient to attacks from one generation to another. Manually applying patches during recovery is a step towards this ideal, but only to a very limited extent—the long duration of the process to discover vulnerabilities and develop appropriate patches provides ample opportunities for attackers to prevail.

An alternative design is to embrace *temporal diversity*, defined to be the diversity among different generations of the same component. Temporal diversity makes it unlikely for different generations of the same component to have the same vulnerability. The work on *proactive obfuscation* at Cornell is looking at this new emerging research direction.

2 Bridging the gap between cryptography and systems

Confidentiality requirements for survivable systems fall on the shoulders of cryptography. The multi-party nature of survivable systems (due to replication) creates practical applications for relatively less-known cryptographic branches such as secret sharing [17, 2], zero-knowledge proofs [8], and secure multi-party computation.

Despite the emergence of systems that successfully incorporated such cryptographic primitives (see [16] for a list of such systems), obstacles remain. Most secure multi-party cryptographic protocols are too expensive to be practical. To make matters worse, composing those cryptographic primitives could be tricky. One well-known example is the parallel or concurrent composeability of zero-knowledge proofs [6]. This is in contrast to the widely used cryptographic building blocks for encryption and digital signatures, which can be relatively easily composed and integrated to a larger and more complicated protocols.

A more fundamental problem lies in the gap between theoretical definitions for security in cryptography and the practical security requirements from real systems. There are often cases where tremendous complexity is added to the protocol in order to achieve provable security, but it is not clear whether the increased complexity translates into security improvements in practice. How to capture security requirements in real systems while still allowing efficient and provably secure solutions continues to be a challenge. The random oracle model [1] is a step in that direction.

3 The power of alliance

In order to gain topological diversity, a survivable system is often distributed over the network and spans multiple administrative domains. The underlying communication network thus becomes both the target of denial-of-service (DoS) attacks and the means to launch such attacks. This poses a new dimension of challenges to survivable systems that is beyond the capability of traditional fault tolerance and cryptography.

The essence of DoS attacks is the over-consumption of system resources by attackers that leaves no resources available for legitimate users. The effectiveness of defense mechanisms such as rate limiting and filtering is limited, especially against distributed DoS attacks, because it becomes almost impossible to distinguish legitimate users from attackers.

A naive over-provisioning of resources to deal with DoS attacks is not cost-effective, but can be made so through an *alliance* formed by a set of systems to defend against DoS attacks. Members promise to contribute resources to help other members when they are under a DoS attacks. The power of alliance hinges on the fact that an alliance jointly has sufficient resources to satisfy the normal loads on its members and to withstand DoS attacks against a small number of its members. An example of this approach was proposed in [20].

For an alliance to be effective against DoS attacks, the alliance must constantly re-allocate resources based on the loads on its members. This is relatively straightforward when members trust each other, and the services provided by members can promptly donate resources and utilize the newly available resources. The problem becomes much harder if the alliance crosses trust boundaries. One interesting future research direction is to study how the prospect of joining an alliance impacts the design of a member.

4 Deception and dynamic defense

"All warfare is based on deception." — Sun Tze

When we move from reliability to survivability and from tolerating faults to combating malicious attacks, the landscape has changed fundamentally: we are now at war with malicious attackers. As a result, our mindset should change accordingly.

The current dismal state of computer/network security can be partly attributed to the fact that we, as defenders, have not embraced the fundamental principles in the art of war: deception. For example, without deception, an attacker can easily search for vulnerabilities using trial-and-error, because known vulnerabilities are either patched or exploitable. A simple application of deception would make much harder the adversary's task of identifying existing vulnerabilities without being detected: instead of patching vulnerabilities, traps are being placed and will be triggered when those vulnerabilities are exploited. Those traps could be used to deter the attackers, track the behavior of attackers, and raise specific alarms that enable fast countermeasures. The idea of using deception to observe how attackers work is essential to projects such as Honeypots [18]. However, most of such tools are for research only and are isolated from the actual systems.

The success of deception hinges on dynamic defense because adversaries learn and adapt quickly. Any static defense is bound to fail over time because defending against every possible vulnerability is impossible for any non-trivial systems—one vulnerability is all it takes for an adversary to succeed. It is worth pointing out that both temporal diversity and the power of alliance manifest the philosophy of dynamic defense.

Incorporating dynamic defense poses new challenges to systems. The dynamism introduced to a system could render certain optimizations infeasible and require new techniques for testing such a constantly-changing system. A system must also balance dynamism in its defense mechanisms with predictability in its services to the users. For mission critical systems, trading off other system properties such as performance for improved survivability is certainly the right choice.

"Those who are able to adapt and change in accord with the enemy and achieve victory are called divine." — Sun Tze

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *The First Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[2] G. R. Blakley. Safeguarding cryptographic keys. In R. E. Merwin, J. T. Zanca, and M. Smith, editors, *Proceedings of the 1979 National Computer Conference*, volume 48 of *AFIPS Conference Proceedings*, pages 313–317, New York, NY USA, September 1979. AFIPS Press.

[3] R. Canetti, S. Halevi, and A. Herzberg. Maintaining authenticated communication in the presence of break-ins. *Journal of Cryptography*, 13(1):61–106, 2000.

[4] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, November 2002.

[5] S. Forrest, A. Somayaji, and D. Ackley. Building diverse computer systems. In *Proceedings of the Sixth Workshop on Hot Topics in Operating Systems*, pages 67–72, Cape Cod, MA, May 1997. Computer Society Press.

[6] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996.

[7] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proceedings of the 19th Annual Conference on Theory of Computing, STOC'87*, pages 218–229, New York, NY USA, May 25–27 1987. ACM.

[8] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18:186–208, 1989.

[9] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, *Advances in Cryptology—Crypto'95, the 15th Annual International Cryptology Conference, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 457–469, Berlin, Germany, 1995. Springer-Verlag.

[10] M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multi-party computation. *Journal of Cryptology*, 13(1):31–60, 2000.

[11] I. Keidar and K. Marzullo. The need for realistic failure models in protocol design. In *Proceedings of International Survivability Workshop*, 2002.

[12] J. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multi-version programming. *IEEE Transactions on Software Engineering*, SE–12(1):96–109, January 1986.

[13] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[14] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[15] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.

[16] F. B. Schneider and L. Zhou. Distributed trust: Supporting fault-tolerance and attack-tolerance. Technical Report TR 2004-1924, Computer Science Department, Cornell University, Ithaca, NY USA, January 2004. Submited for publication.

[17] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[18] L. Spitzner. *Honeypots: Tracking Hackers.* Addison-Wesley Publishing Company, September 2002. ISBN: 0321108957.

[19] J. H. Wensley. Further comments on "the reliability of periodically repaired $n-1/n$ parallel redundant systems. *IEEE Transactions on Computers*, 34(11):1068, 1985.

[20] J. Yan, S. Early, and R. Anderson. The XenoService—a distributed defeat for distributed denial of service. In *Proceedings of the CERT Information Survivability Workshop (ISW 2000)*, Boston, MA USA, October 2000.