The evolution of computational systems has always trended in the direction of increasing complexity. While early machines were designed for simple sequential computations, modern computational systems routinely operate concurrently in distributed environments connected by adversarial and unreliable networks. Ensuring safety and security of distributed computational systems requires novel programming abstractions and formal (reasoning) methods that enable programmers overcome the overwhelming complexity inherent to the design of these systems.

> *My research attempts to make foundational advances in programming languages and formal methods with aim of making provably-safe distributed systems development accessible to mainstream software developers.*

Formal methods have seen increasing adoption in mission-critical systems in the recent years. DARPA's HACMS program, for example, demonstrated the effectiveness of formal verification in securing unmanned aerial vehicles [4]. However, their widespread adoption remains a challenge due to the steep learning curve, integration complexity, and scalability concerns associated with formal methods. Bridging the gap between theoretical advances and practical deployment in industry requires research into more accessible tools, automated reasoning techniques, and programming abstractions that can seamlessly integrate with existing development workflows. My work is motivated by these challenges and aims to make formal methods more practical and impactful for real-world software systems. My research program involves three major thrusts, numbered R_{1-3}, to address scalability, generality, and usability challenges associated with formal methods. I will next summarize each research thrust, highlight significant contributions, and present their broader impacts and applications.

R_1 Scaling Formal Verification to Real Systems

Programming safe and reliable distributed systems is extremely hard [21, 14, 8]. The complexity stems from having to simultaneously reason about several failure modes, such as reordered and dropped messages, network partitions, and node crashes. Complex combinatorial reasoning involving low-level events and high-level application logic is hard for humans and machines alike, making it challenging to apply formal methods at scale. Consequently, formal certification of safety and reliability guarantees of distributed systems remains out of reach for most application developers. Any formal verification that is currently done in the industry is carried out at the level of distributed protocol specifications, e.g., using TLA+ [13] or P [5], with no way to translate the guarantees to implementations.

Several decades of research into programming languages has shown that modularity and abstraction is often the key to overcoming programming complexity. The abstractions typically used to program distributed systems today expose asynchronous communication primitives and require the explicit reasoning about the dynamics of the network state in relation to faults. To scale formal verification to real-world systems implementations, we need novel programming abstractions that decompose the complexity of fault tolerance and protocol logic into loosely-coupled *modules* and *services* amenable to independent specification and verification.

> *Thrust 1 focuses on developing novel programming abstractions that enable modular specification, verification, and implementation of distributed systems.*

The primary motivation for modular decomposition in our case is automated verification as opposed to efficient implementation. To be amenable to SMT-aided automated reasoning, verification

conditions should be encodable in a a decidable logic, which requires deliberate design of modules and interfaces. As a part of ongoing work in Thrust 1, we are developing language-embedded specification and reasoning frameworks that reduce the manual effort and creative inputs involved in this process. The broadest impact will be on mainstream application programmers, who may not have the expertise to build mission-critical distributed systems. Our research will help programmers overcome the complexity of this task by making formal verification and machine-assisted reasoning readily accessible.

Significant Contributions Our early work in Thrust 1 [9, 10, 11] focused on development of *Mergeable Replicated Data Types* (MRDTs) – a high-level distributed programming abstraction that guarantees fault tolerance by design. We built a distributed runtime for MRDTs, called Quark [11], that orchestrates distributed computation in a way that is guaranteed to converge. This is in contrast to the state-of-the-art RDT abstractions, which require the programmer to prove the convergence of a computation explicitly [28]. Quark has allowed us to uncover novel performance tradeoffs in distributed applications and collect strong experimental evidence demonstrating the viability of high-level programming abstractions for *coordination-free* distributed computations. Theoretically, RDTs are a high-level abstraction of a weakly-consistent replicated state machine (RSM), denoted ♦ RSM. While ♦ RSMs are often useful, safety-critical distributed systems need strongly-consistent (■) RSMs, whose implementations are known to be notoriously complex. In our recent work [15], we have shown that ♦ RSMs can be extended with modules of inter-replica coordination to obtain ■ RSMs, resulting in a modular approach that drastically reduced the overhead of safety verification. We implemented this technique in a verification framework called Super-V and used it to automatically verify an industry-strength implementation of Raft log replication protocol [22]. We believe this is a notable achievement considering that verification efforts of similar nature required 2-3 orders of magnitude more manual annotations [31]. Our results will soon be presented at SIGPLAN OOPSLA 2025 conference.

Funding and Project Management The primary source of funding has thus far been my startup grant. An Amazon Research Award providing $50,000 to this work had to be declined on insistence of CU's office of contracts and grants (OCG). An NSF Small proposal titled *A Modular Approach to Practical Distributed Systems Verification* is currently pending review. The proposed budget is $480,110 to be spent over three years. The project team includes Mr. Nicholas Lewchenko, a PhD student advised by me. Mr. Lewchenko will graduate in Spring 2026 and will be replaced by a new graduate student yet to be admitted.

R_2 Generalizing Formal Verification to Adversarial Networks

Like the prior work on distributed systems verification [30, 31, 27], Thrust 1 focuses on *crash faults* such as node crashes and network partitions. In conventional data center-based deployments, exceptional behavior is almost exclusively due to crash faults. However, distributed systems now increasingly operate the public internet, where they are used to implement novel decentralized applications, such as blockchain cryptocurrencies [20, 3] and federated social networks [18, 29]. A decentralized application is effectively a distributed protocol designed to operate on a public network. The early successes of decentralized applications and the push towards local-first software [12] are expected to accelerate the development of a novel class of distributed protocols designed to execute on internet-native distributed systems. While network partitions remain a major concern on public networks, more serious concern is the presence of adversaries who can actively subvert protocol executions towards unsafe or unproductive states *even* when all the protocol participants are honest (i.e., non-byzantine). Ensuring the safety of distributed systems and the privacy of their

users on adversarial networks requires synthesizing reasoning and verification techniques from Cryptography and Distributed Systems.

> *The focus of Thrust 2 is to develop a unified framework that combines cryptographic proofs of security with formal verification of distributed protocols.*

The goal is to help developers obtain strong guarantees of safety and privacy in adversarial environments. Like in Thrust 1, we seek modular abstractions that allow compositional specification and verification of protocol components, but now with explicit modeling of adversarial actions and cryptographic primitives. In our ongoing work, we are developing frameworks that enable automated reasoning about both protocol correctness and cryptographic soundness, aiming to reduce the manual effort required for end-to-end verification. This approach will empower developers to build robust decentralized applications that are resilient to both crash faults and active attacks, facilitating broader adoption of secure distributed systems in practice.

Significant Contributions In [17], we introduced a novel formal model of cryptography and an associated probablistic relational verification framework. In formal verification of cryptographic protocols, a network attacker is modeled in one of the two fundamentally different ways: (a). a *symbolic* or *Dolev-Yao* attacker defined in terms of what the attacker can do [6], and (b). a *computational* attacker defined in terms of what the attacker can not do [2, 1]. Correspondingly, two standard models of cryptography – *Symbolic* and *Computational* – have emerged. While the former is simpler and amenable to symbolic reasoning, it underapproximates attacker capabilities, hence misses several classes of attacks. Conversely, the latter is more general and precise, but requires arduous manual proofs. In [17], we introduced a probabilistic model of cryptography that combines the strengths of both: it is more precise than the symbolic model (hence covers more classes of attacks) and more amenable to automation than the computational model. We implemented the probabilistic model in an SMT-aided verification framework called Waldo. Waldo is released publicly with an open source license, and was used to identify subtle privacy issues in the draft proposal for TLS Encrypted Client Hello (ECH) extension [25]. We believe this is a significant milestone considering that TLS is the protocol powering secure HTTP connections for the entire internet, hence any privacy issues are bound to have adverse consequences.

Funding and Project Management The primary source of funding has thus far been my startup grant. My NSF CAREER proposal titled *Verifying Safety and Privacy of Distributed Systems on Adversarial Networks* is currently under review. The proposed budget is $714,519 to be spent over five years for the activities in Thrust 2. In addition, we have partnered with Psiphon Inc [24] to provide inputs to DARPA PM Mr. Michael Lack on formulating a new program on privacy verification on internet. The project team includes PhD students Mr. Kirby Linvill and Mr. Sai Aka, both advised by me.

R_3 Making Formal Verification Accessible to Non-Experts

A significant progress has been made in the formal methods community towards automating symbolic reasoning for complex systems. However, the tools and languages developed often require a specialized skill set to write precise mathematical specifications and proofs, which is often out of reach for mainstream application programmers. To make formal methods an accessible and routine part of the software development lifecycle, there is a need for novel techniques and tools that flatten the learning curve and lower the barrier to entry. The recent progress in Large Language Models (LLMs) provides an exciting opportunity to (a). leverage natural languages as interfaces

to formal verification tools, and (b). use generative AI to guide the proof search process. There are also notable new inventions in programming language techniques, such as the Incorrectness Logic and Goal-directed abstract abstract interpretation, which require minimal user inputs to drive formal verification. In Thrust 3, our goal is to leverage these developments to drastically improve the usability of formal methods, particularly in the context of distributed systems.

> *Thrust 3 focuses on developing symbolic and neuro-symbolic reasoning techniques to automate formal verification for distributed systems.*

Significant Contributions In [16], we proposed a novel partial-function semantics for higher-order functional programs that makes them amenable to decidable encoding in SMT solvers. Partial functions underapproximate total functions, which makes formal verification unsound in general. Our key contribution is a set of sufficient conditions under which the underapproximation is actually sound. We developed a language extension to Rust, called Λ_{EPR}, that lets non-expert programmers use SMT solvers to automatically verify deep semantic properties of higher-order functional programs. Another notable contribution is the DISSPROVE verification framework [7] for distributed protocols that leverages a novel goal-directed backwards analysis we introduced in our earlier work [19] to eliminate the need for inductive invariants (for a class of protocols). Identifying the right inductive invariants is often hardest step in formal verification of distributed protocols [23]. By short-circuiting this process, DISSPROVE makes it feasible for non-experts to build provably-safe distributed systems.

Ongoing Work In the above, we have only considered systems with discrete dynamics. Our ongoing work extends formal verification to systems with continuous dynamics where state transitions are defined in terms of ordinary differential equations (ODEs). The complex structure of the ODEs often makes it expensive to perform computations such as simulations and probabilistic inference. In our ongoing work, we are building a DSL for probabilistic programming with ODEs coupled with a verified optimizing compiler. The compiler orchestrates a series of semantics-preserving transformations, such as latent variable elimination, to automatically transform ODEs into a form amenable to efficient inference. This allows domain experts to focus on specifying system dynamics without concerning themselves with efficient implementations. In another thread of ongoing work, we showed that purely neural approaches, including the state-of-the-art LLMs, perform poorly on program synthesis tasks [26]. To improve the effectiveness of LLM-driven program synthesis, we designed a semantics-constrained decoding technique that uses conventional symbolic techniques to prune unproductive inference paths. The early results are promising and we are working on developing a stand-alone tool to apply our insights at scale.

Funding and Project Management The verified ODE transformation work is funded by the NSF Formal Methods in the Field (FMitF) program from 09/01/2024 to 08/31/2028 through award number: 2422136, amount: $875,000, titled: *FMitF: Track I: Verified Probabilistic Programming for Hybrid Systems*. An Amazon Research Awards (ARA) proposal titled *Program Synthesis with Syntax-Aligned Language Models* is currently under review. The project team includes Mr. Oscar Bender-Stone, an undergraduate student on Discovery Learning Apprenticeship (DLA) program, along with the PhD students Mr. Christian Fontenot, Mr. Nicholas Lewchenko, Mr. Kirby Linvill, and Ms. Manasvi Parekh.

References

[1] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006. URL https://doi.org/10.1007/11761679_25.

[2] Bruno Blanchet and Charlie Jacomme. CryptoVerif: a computationally-sound security protocol verifier. Technical Report RR-9526, Inria, October 2023. URL https://inria.hal.science/hal-04253820.

[3] Vitalik Buterin. Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform. *Distributed online*, 2014. URL https://ethereum.org/en/whitepaper/. Originally published on his blog and later widely distributed.

[4] Defense Advanced Research Projects Agency (DARPA). High-Assurance Cyber Military Systems (HACMS). https://www.darpa.mil/research/programs/high-assurance-cyber-military-systems, 2025. Accessed: 2025-08-12.

[5] Ankush Desai, Vivek Gupta, Ethan Jackson, Shaz Qadeer, Sriram Rajamani, and Damien Zufferey. P: Safe asynchronous event-driven programming. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, page 321–332, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320146. doi: 10.1145/2491956.2462184. URL https://doi.org/10.1145/2491956.2462184.

[6] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. doi: 10.1109/TIT.1983.1056650.

[7] Christian Fontenot, Gowtham Kaki, and Bor-Yuh Evan Chang. Dissprove: Goal-directed verification of parametrized affine actor systems. Under submission to ACM SIGPLAN POPL, 2026.

[8] Yu Gao, Wensheng Dou, Feng Qin, Chushu Gao, Dong Wang, Jun Wei, Ruirui Huang, Li Zhou, and Yongming Wu. An empirical study on crash recovery bugs in large-scale distributed systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, page 539–550, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355735. doi: 10.1145/3236024.3236030. URL https://doi.org/10.1145/3236024.3236030.

[9] Gowtham Kaki, Swarn Priya, KC Sivaramakrishnan, and Suresh Jagannathan. Mergeable replicated data types. *Proc. ACM Program. Lang.*, 3(OOPSLA), oct 2019. doi: 10.1145/3360580. URL https://doi.org/10.1145/3360580.

[10] Gowtham Kaki, KC Sivaramakrishnan, and Suresh Jagannathan. Version Control Is for Your Data Too. In Benjamin S. Lerner, Rastislav Bodík, and Shriram Krishnamurthi, editors, *3rd Summit on Advances in Programming Languages (SNAPL 2019)*, volume 136 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:18, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-113-9. doi: 10.4230/LIPIcs.SNAPL.2019.8. URL http://drops.dagstuhl.de/opus/volltexte/2019/10551.

[11] Gowtham Kaki, Prasanth Prahladan, and Nicholas V. Lewchenko. Runtime-assisted convergence in replicated data types. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2022, page 364–378, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392655. doi: 10.1145/3519939.3523724. URL https://doi.org/10.1145/3519939.3523724.

[12] Martin Kleppmann, Adam Wiggins, Peter van Hardenberg, and Mark McGranaghan. Local-first software: You own your data, in spite of the cloud. Onward! 2019, page 154–178, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369954. doi: 10.1145/3359591.3359737. URL https://doi.org/10.1145/3359591.3359737.

[13] Leslie Lamport. The temporal logic of actions. *ACM Trans. Program. Lang. Syst.*, 16(3):872–923, May 1994. ISSN 0164-0925. doi: 10.1145/177492.177726. URL https://doi.org/10.1145/177492.177726.

[14] Tanakorn Leesatapornwongsa, Mingzhe Hao, Pallavi Joshi, Jeffrey F. Lukman, and Haryadi S. Gunawi. Samc: Semantic-aware model checking for fast discovery of deep bugs in cloud systems. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, page 399–414, USA, 2014. USENIX Association. ISBN 9781931971164.

[15] Nicholas V. Lewchenko, Gowtham Kaki, and Bor-Yuh Evan Chang. Bolt-on strong consistency: Specification, implementation, and verification. *Proc. ACM Program. Lang.*, 9(OOPSLA1), April 2025. doi: 10.1145/3720502. URL https://doi.org/10.1145/3720502.

[16] Nicholas V. Lewchenko, Gowtham Kaki, and Bor-Yuh Evan Chang. Effectively-propositional higher-order functional programming. Under revision process at OOPSLA, 2025.

[17] Kirby Linvill, Gowtham Kaki, and Eric Wustrow. Verifying indistinguishability of privacy-preserving protocols. *Proc. ACM Program. Lang.*, 7(OOPSLA2), October 2023. doi: 10.1145/3622849. URL https://doi.org/10.1145/3622849.

[18] Mastodon Documentation. Technical overview — Mastodon documentation. https://docs.joinmastodon.org/dev/overview/, October 2024. Accessed on Mastodon Developer Documentation.

[19] Shawn Meier, Sergio Mover, Gowtham Kaki, and Bor-Yuh Evan Chang. Historia: Refuting callback reachability with message-history logics. *Proc. ACM Program. Lang.*, 7(OOPSLA2), October 2023. doi: 10.1145/3622865. URL https://doi.org/10.1145/3622865.

[20] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Distributed by email to the Cryptography Mailing List at metzdowd.com*, October 2008. URL https://bitcoin.org/bitcoin.pdf. Whitepaper.

[21] Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. How amazon web services uses formal methods. *Commun. ACM*, 58(4):66–73, mar 2015. ISSN 0001-0782. doi: 10.1145/2699417. URL https://doi.org/10.1145/2699417.

[22] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, page 305–320, USA, 2014. USENIX Association. ISBN 9781931971102.

[23] Oded Padon, Kenneth L. McMillan, Aurojit Panda, Mooly Sagiv, and Sharon Shoham. Ivy: Safety Verification by Interactive Generalization. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 614–630, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4261-2. doi: 10.1145/2908080.2908118. URL http://doi.acm.org/10.1145/2908080.2908118.

[24] Psiphon Inc. Psiphon: Uncensored Internet Access. https://psiphon.ca/, 2025. Official website for the internet censorship circumvention tool.

[25] E. Rescorla, K. Oku, N. Sullivan, and C.A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-15, IETF Secretariat, October 2022. URL https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-15.

[26] Richard Roberson, Gowtham Kaki, and Ashutosh Trivedi. Analyzing the effectiveness of large language models on text-to-sql synthesis, 2024. URL https://arxiv.org/abs/2401.12379.

[27] Ilya Sergey, James R. Wilcox, and Zachary Tatlock. Programming and proving with distributed protocols. *Proc. ACM Program. Lang.*, 2(POPL), December 2017. doi: 10.1145/3158116. URL https://doi.org/10.1145/3158116.

[28] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-Free Replicated Data Types. In Xavier Défago, Franck Petit, and Vincent Villain, editors, *Stabilization, Safety, and Security of Distributed Systems*, volume 6976 of *Lecture Notes in Computer Science*, pages 386–400. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24549-7. doi: 10.1007/978-3-642-24550-3_29.

[29] The Bluesky Team. Federation Architecture Overview. https://bsky.social/about/blog/5-5-2023-federation-architecture, May 2023. Bluesky Blog Post.

[30] James R. Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Thomas Anderson. Verdi: A Framework for Implementing and Formally Verifying Distributed Systems. In *PLDI*, pages 357–368, 2015.

[31] Doug Woos, James R. Wilcox, Steve Anton, Zachary Tatlock, Michael D. Ernst, and Thomas Anderson. Planning for change in a formal verification of the raft consensus protocol. CPP 2016, page 154–165, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341271. doi: 10.1145/2854065.2854081. URL https://doi.org/10.1145/2854065.2854081.