Chris Beck

Contact Information

mobile: (upon request)
e-mail: beck.ct@gmail.com

Mission

I enjoy thinking about complex problems and solving them from first principles, especially when performance and correctness are both critical.

I was trained as a Theoretical Computer Scientist, with a focus in computational complexity theory. I now work as a research engineer and a cryptographer, with a strong background in high-performance low-level systems programming. I have expert level knowledge in Private Information Retrieval and Oblivious RAM, and familiarity with a few ZK-proving systems.

I like working with smart, ambitious people who have clear goals and want to work on things that will impact the world positively.

Research Interests

Computational Complexity, Pseudorandomness, Cryptography, Algorithms, Combinatorics

Education

Princeton University, Princeton, New Jersey, USA

Doctor of Philosophy: Computer Science **2009 – 2014**

California Institute of Technology, Pasadena, California, USA

Bachelors of Science with Honor: Mathematics **2005 – 2009**

Bachelors of Science with Honor: Computer Science **2005 – 2009**

Honors and Awards

Wu Prize for Excellence, 2013

Simons Award for Graduate Students in Theoretical Computer Science, 2012-2014

NSF GRFP Honorable Mention, 2009

The G. Wallace Ruckert '30 Fellowship, 2009

Professional Experience

Fire Digital Inc., San Francisco, California, USA

Founder, Chief Technology Officer **July 2024 – present**

I cofounded and was responsible for engineering at Fire Digital, a digital assets service provider which brokered 10M USD in volume of Cryptocurrency trades on behalf of high-net worth individuals in California. Highlights of this work included managing a team of 7 engineers and designers, designing and implementing a custodial system that meets regulatory requirements, and designing and implementing high-frequency trading algorithms in Rust.

MobileCoin Inc., San Francisco, California, USA

Research Engineer, Cryptographic Engineer **Nov 2018 – May 2024**

MobileCoin Core Engineering, Architect of MobileCoin Fog, Cryptogaphy Engineering Manager

I was the fifth engineer hired at MobileCoin. Over my time here, my responsibilities have often shifted according to the needs of the project. I have often taken responsibility for mission critical tasks. Some highlights, in roughly reverse chronological order, include:

Privacy-preserving atomic swap protocol

- MobileCoin wanted to have a privacy-preserving DEX on our chain, however, creating private smart contracts is a difficult research problem and the time horizon exceeded what was relevant from a business perspective. (We tabled this discussion in Q2 2021).
- In Q2 2022 I conceived of a new approach. I designed and implemented a privacy-preserving atomic swap protocol which works on our chain without creating smart contracts. This implementation took a few weeks. This is a peer-to-peer protocol on-top of our L1 chain. It adds no appreciable delay to a normal MobileCoin transaction.
- The initial version was specified in MCIP 31. We extended the initial design to support partial-fill trades, in MCIP 42. Implemented the network in 2023. (https://github.com/mobilecoinofficial/deqs)

MobileCoin Fog

- In 2018, MobileCoin faced an existential problem around how we can integrate with Signal messenger app whilst meeting Signal's privacy requirements in a scalable way.
- Leadership imagined that it would work like Monero – the phones would download and view-key scan the entire blockchain. However, Moxie Marlinspike, the CEO of Signal, rejected this.
- They then proposed pushing this work onto the server – SGX enclaves would view-key scan on behalf of all of the users. However, this doesn't scale in an affordable way to millions of users.
- Some engineers proposed "sharding" the users, to enable a trade-off between privacy and efficiency. Moxie rejected all forms of this.
- I lead a research effort to find a way that the users can find their transactions, without revealing any information to Signal or creating massive computational work on server or client side.
- I conducted an overview of the state of the art in Private Information Retrieval, and determined and Oblivious RAM inside of SGX enclaves may be a viable path forwards.
- I proposed an architecture for the MobileCoin Fog service based on experimental Oblivious RAM implementations, and provided order of magnitude estimates as to the scaling needs and costs.
- I wrote a 40 page document explaining this proposal for Signal, which they accepted.
- I implemented a version of ECIES public key encryption based on the Ristretto group, used for sending encrypted messages to the enclave.
- I proposed and implemented modifications to the MobileCoin public address format, transaction format and transaction builder to be compatible with this new system.
- I proposed and implemented a "key-exchange random number generator" used to privately index each user's transactions in a way that only they can detect.
- I proposed and implemented a service which loads all the transaction data into an Oblivious RAM and makes it searchable by the random values generated thusly.
- This novel system was patented and a journal publication is forthcoming.
- I developed this system and worked to support demos of our technology.
- This system has been in production for 2.5 years, having launched to production supporting millions of Signal users worldwide in February 2021, without significant incident.

MobileCoin Consensus Network

- I played a significant role in designing MobileCoin's consensus validator enclave.
- Implementation of privacy preserving proofs of membership for transaction validation.
- Developed low-level SGX infrastructure (allocator, mutex, etc.) which we use instead of the rust standard library, to reduce attack surface area. (The normal rust standard library cannot be used because system calls to the OS undermine the SGX security model.)
- Developed a structured hashing scheme compatible with protobuf-style Schema Evolution. This is used for all hashing and digital signatures within MobileCoin, and for the blockchain itself.
- Helped to improve the performance of locking schemes used in the consensus nodes.
- In later years, wrote numerous MCIPs and designed numerous features for MobileCoin, including encrypted memos, confidential token ids, minting and burning protocols.

Tesla Inc., Palo Alto, California, USA

Senior Software Engineer **Oct 2017 – Nov 2018**

Autopilot Software Infrastructure.

I wrote C++ code that runs in the car and is used by critical software components like Vision, Camera, Can-bus, to communicate. Many such components are organized in the car computer as independent linux processes, using Posix shared memory.

High-performance lock-free shared memory ringbuffers

- Several implementations were given to support tradeoffs: (strong wait-free guarantee, multiple producers or consumers, fixed or variable-length payloads)
- These have various applications in the car for sharing critical data between processes, or logging activities of processes / changing of critical data
- Designed to be used in shared memory for interprocess communication, initialized lazily / on-demand, portable but optimized for x86 and arm architectures
- Implemented a clean "single-step" deterministic test framework for fuzzing these

Implementing a new interface and back-end for glog-style text logging

- Google's logging library was found to be making dynamic memory allocations and writing to pipes, not real-time friendly according to our guidelines. I was assigned to fix this.
- I replaced the entire glog interface and back-end, which writes over shared memory ring buffers in order not to block the critical path in real-time tasks.
- Our interface avoided several inefficiencies e.g. when several things are streamed at once into logging, we compute in advance before formatting how much space is needed to log everything, then we make a checkout in ringbuffer and format there without copying.
- We used expression templates rather than conventional ostream API to ensure that we can see all the arguments without copying any of them, before having to handle any one of them.
- This resolved frequent TOI interventions due to watchdog timeouts in developer builds

Implementing modern (real-time / embedded-friendly) C++ utility libraries

Because many autopilot tasks have real-time critical safety requirements, many language features like exceptions and dynamic memory cannot be used at all while the car is moving. I implemented equivalent versions of many STL classes that can be used safely in real-time tasks.

- std::variant, std::function, std::ostream
- Constexpr perfect hash maps

Programming Proficiencies

Languages: Rust, C++, C, Python, Bash scripting, Lua, Matlab
Web Technologies: HTTP, GRPC, TLS, SQL, LMDB, Posix shared memory, Intel SGX

Open Source Contributions

visit_struct (Lead developer) https://github.com/cbeck88/visit_struct
A tiny library that provides for struct-field reflection in C++11. It is portable to many versions of gcc, clang, and msvc, and many github users tell me they have used it in production. As of 2018 it is used in Tesla autopilot code for certain code-gen tasks.

strict_variant (Lead developer) https://github.com/cbeck88/strict_variant
A simple and efficient type-safe union for C++11, which is embedded / real-time friendly, meaning it is very easy to use it in a way that avoids C++ exceptions and dynamic allocations. As of 2018 it is used in Tesla autopilot code.

spirit_po (Lead developer) https://github.com/cbeck88/spirit_po
A library that parses the gettext po format, and can replace the use of GNU libintl in projects that use the GNU gettext system for internationalization of software. spirit_po is written using the boost::spirit parser framework, it is in total about 900 lines of code. It has been used in the Battle for Wesnoth project for several years now.

Academic Career

Institute for Advanced Study, Princeton, New Jersey, USA

Postdoctoral Scholar **Sept 2014 – August 2016**

Research in computational complexity theory, especially time space tradeoffs and pseudorandomness.

Selected Publications

P. Beame, C. Beck, R. Impagliazzo. Time-Space Tradeoffs in Resolution: Superpolynomial Lower Bounds for Superlinear Space. *Proceedings of the 44th Annual ACM Symposium on Theory of Computing* (STOC 2012). **Also in special issue of SIAM Journal on Computing 2016 45:4, 1612-1645 .**

C. Beck, R. Impagliazzo, S. Lovett. Large Deviation Bounds for Decision Trees and Sampling Lower Bounds for AC0-circuits. *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science* (FOCS 2012).

C. Beck, J. Nordström, B. Tang. Some Tradeoffs in Polynomial Calculus. *Proceedings of the 45th Annual ACM Symposium on Theory of Computing* (STOC 2013).

C. Beck, R. Impagliazzo. Strong ETH Holds for Regular Resolution. *Proceedings of the 45th Annual ACM Symposium on Theory of Computing* (STOC 2013).

Academic References

Additional contact information available upon request.

Professor Sanjeev Arora
Professor of Computer Science
Princeton University
Princeton, New Jersey, USA
phone: *available on request*
e-mail: arora@cs.princeton.edu

Professor Russell Impagliazzo
Professor of Computer Science
University of California, San Diego
San Diego, California, USA
phone: *available on request*
e-mail: russell@cs.ucsd.edu

Professor Shachar Lovett
Asst. Professor of Computer Science
University of California, San Diego
San Diego, California, USA
phone: *available on request*
e-mail: slovett@math.ias.edu

Professor Paul Beame
Professor of Computer Science
University of Washington
Seattle, Washington, USA
phone: *available on request*
e-mail: beame@cs.washington.edu