Chris Beck

Contact Information

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

Research Interests

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

MobileCoin Inc., San Francisco, California, USA

Research Engineer, Cryptographic Engineer **Nov 2018 – present**

MobileCoin core engineering, Architect of MobileCoin Fog, Fog Team Lead

At the beginning of the startup, my responsibilities shifted often according to the needs of the project. Some highlights related to launching SGX consensus network include:

Groundwork for SGX Infrastructure

- Developed replacements for rust standard library inside of SGX, including allocator and mutex
- The standard library cannot be used in SGX because system calls to the OS undermine the security model
- Built out a rust environment on top of which transaction logic was implemented
- This infrastructure was ultimately shipped in production

Serialization infrastructure for enclaves and blockchain

- Built infrastructure and maintained patches to ensure that well-understood formats with good security properties would also build in a no-standard-library configuration in order to be compatible with our SGX enclaves
- Also responsible for using this in the implementation of the MobileCoin ledger database using lmdb
- Collaborated with many engineers in this process

Developed a structured hashing scheme compatible with protobuf-style Schema Evolution

- Blockchains require hashing structured data in a non-malleable way
- Developed the solution in 'mc-crypto-digestible' which uses rust proc macros to walk structures
- Marshalls structures into the 'merlin' API developed by dalek-cryptography team
- The design of this hashing scheme enables us to extend our structures with optional elements, just as protobuf, without changing the hashes of old objects.
- This innovation greatly simplifies the process of adding new features to the blockchain without breaking old code, and has enabled MobileCoin to role out new features faster.
- This scheme is now used for most hashes and digital signatures in the MobileCoin software ecosystem.

Within the first year, my most important responsibility was around the integration of MobileCoin with Signal. The major technical challenge here is that privacy coin technologies make it very hard to

mobile device to the cloud, because this isn't compatible wiht our threat model and violates Signal's requirements. Coming up with a new technology to solve these problems was one of the most critical challenges facing MobileCoin.

Proposal to build MobileCoin Fog (2018-2019)

- Researched Private Information Retrieval literature and Signal Contact Discovery
- Determined that Oblivious RAM may be a viable technology when used in conjunction with SGX
- Proposed a way for Fog to discover where to route transactions without revealing information outside of SGX, using public key encryption
- Proposed a way for Fog to index the transactions using a random number generator driven by cryptographic key exchange
- Proposed a way for Fog to have high-availability by sharing secrets across mutually attested SGX enclaves
- Proposed a way for Fog to further support phones in conducting private balance checks with low bandwidth
- Conducted order of magnitude estimates to predict the efficiency of the system before builing it
- Conducted a security analysis to demonstrate that the system resists active attacks by the node operator
- Lobbied internally within the company to establish confidence that we can deliver the system
- Wrote a very detailed technical proposal for Moxie Marlinspike, which was accepted after negotiation

Architecture and Demos of MobileCoin Fog (2019-2020)

- Adopted a microservices architecture for the system
- Developed a client for demonstration purposes that performs balance checks and sends transactions
- Worked closely with mobile developers to implement the Fog protocol in mobile SDKs and pin down APIs
- Collaborated to develop interactive demos for investors using the web and mobile phones
- Implemented cryptographic libraries for public key encryption using the Ristretto elliptic curve
- Implemented cryptographic libraries for key-exchange-seeded random number generators
- Offered technical leadership as our engineering team moved core functionality into enclaves, and established sound enclave APIs
- Developed a comprehensive test plan for the system

Prototypes for Oblivious RAM (2020)

- Developed one, and then a second prototype for Oblivious RAM in SGX based on the Path ORAM and ZeroTrace papers
- Designed the system with several layers of interfaces, abstracting storage away from algorithm, with a view towards Circuit ORAM algorithm
- Created a protocol for the enclave to store encrypted authenticated data using untrusted storage
- Achieved pen-and-paper predictions for performance
- Identified CMOV operation as a CPU bottleneck, and wrote inline assembly using AVX2 instructions to accelerate system performance by 4x

Partner Support (2020-2021)

- Took frequent-calls with Signal to answer questions and speak to concerns
- Frequently made late stage changes to the design to accommodate new requirements
- Redesigned key exchange system to simplify operations and reduce the need for interaction with mobile devices
- Developed a novel hash table that allows an oblivious insert-or-modify operation
- Worked without outside cryptographers to build confidence in the modified system
- Designed and developed an extension of MobileCoin and Fog called "Recoverable Transaction History", primarily to meet Signal's requirements

Production-Readiness of MobileCoin Fog (2021)

- Moved final Oblivious RAM implementation into all SGX enclaves for stress testing before production
- Worked with engineers and SREs to make the system ready for practical deployment scenarios
- Delegated segments of the remaining implementation issues to other engineers effectively in order to hit deadlines
- Finalized data model for the system's database
- Worked with external auditors to build confidence in the security of the system

My ongoing responsibilities include leading the Fog team, continuing to improve performance and reliability of the system, and researching new initiatives for the company.

Tesla Inc., Palo Alto, California, USA

Senior Software Developer **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. My projects and responsbilities have included:

Implementing a series of 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 "glog" 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 Skills

Rust, C++, Python, Lua, Matlab, Bash scripting

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