Ethan Cecchetti – Research Statement

Modern applications are increasingly structured as a composition of independent modules interacting in complex ways. These applications are difficult to secure, especially when the composed modules do not trust each other. The most extreme examples of such architectures—smart contract systems—have lost around $1 billion to numerous attacks since July 2016, when an attacker stole 25% of the funds in the Decentralized Autonomous Organization (DAO)—an Ethereum contract holding 15% of all coins on Ethereum at the time.[1] In addition to their immense financial cost, these attacks serve as warnings for systems like web pages with interacting JavaScript libraries and large service-based cloud applications, which lack the direct control of financial assets that make smart contracts an ideal target, but share the same modular structure with complex trust relationships. My research focuses on understanding and achieving security in such composed systems.

My work notably highlights the need for compositional analyses, which can prove guarantees about an entire system by composing separate analyses of its constituent parts. I have shown how to prevent the entire class of attacks exposed by the DAO using compositional guarantees [8, 9], and how a lack of composition can make security prohibitively expensive in real applications like public decentralized storage networks [7]. Existing analyses are, unfortunately, inadequate. Universally Composable (UC) security [4] and similar techniques [3, 20] ensure that a composition of cryptographic protocols is as secure as the composition of their specifications, but those specifications may subtly leak information and combine in unexpectedly dangerous ways. Programming languages techniques can provide compositional guarantees to specifications, but often assume unrealistically simple computational models.

My research attacks these problems from three main angles. First, I define and enforce new security notions to prevent costly attacks exploiting complex interactions between distrusting modules [5, 8, 9]. Second, I build primitive constructs to securely enable previously-missing functionality needed by practitioners, using new security definitions to clarify applications' security requirements [6, 7, 21]. Finally, I bring my expertise to collaborations, proving secure practical system designs [11] and grounding theoretical security work in real problems [14]. Moving forward, I aim to continue solving real-world security problems in decentralized systems, which will require expanding and combining techniques for compositional reasoning, while also working with colleagues in other disciplines.

New Security Notions for Distrusting Module Systems

Applications composed of mutually distrusting modules admit novel types of attacks requiring new notions of security. One prominent example is reentrancy, where one module calls a second and the second unexpectedly calls back into the first before returning, potentially causing the first to behave improperly. Two reentrancy exploits, attacking the DAO[1] and Uniswap[2], have cost Ethereum smart contracts nearly $100 million since July 2016. Prior defenses [e.g., 10, 12, 16] attempt to protect individual contracts, ignoring the general problems posed by reentrancy—which can occur in any system with user-supplied callbacks—and applications composed of multiple contracts. But composed applications are a major concern! The Uniswap attack used a reentrancy vulnerability created by an unexpected interaction between two otherwise-benign contracts to steal over $25 million.[2]

My work [8, 9] identifies the fundamental role of *trust* in reentrancy; an attack can occur when a *trusted* module calls *untrusted* code that then calls trusted code again, even if the second call goes to a different module. I use information flow control (IFC), a programming language-based technique designed to reason compositionally about influence in decentralized systems, to lay out the first trust-based formulation of reentrancy and reentrancy security. IFC ideas result in definitions that precisely capture the desired security,

[1] `https://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/`

[2] `https://medium.com/@peckshield/uniswap-lendf-me-hacks-root-cause-and-loss-analysis-50f3263dcc09`

but prior IFC schemes fail to secure reentrancy. I thus develop a lock-based enforcement mechanism and prove that it achieves compositional reentrancy security, even in the presence of unknown code.

Even with added reentrancy protections, existing IFC tools—which track both confidentiality and integrity—fall short of analyzing real applications. IFC systems traditionally enforce *noninterference*, the extremely strong guarantee that a program's public (or trusted) outputs remain completely independent of any secret (or untrusted) inputs. Yet real systems fail this requirement. For example, a password checker must take a guess from an untrusted user, compare it to a secret, and produce a public and trusted result depending on both the secret and untrusted inputs. Securing practical applications requires secure ways of endorsing untrusted inputs and declassifying secret-derived outputs.

Several definitions of secure declassification exist [e.g., 15, 17–19], but secure endorsement was relatively unexplored, leaving applications that endorse data without meaningful security guarantees. I address this shortcoming with Nonmalleable Information Flow (NMIF) [5], a semantic condition that rules out attacks that exploit endorsement of secret data to trick a system into accepting its own secrets as if they were user-generated input. NMIF, along with a proved-correct enforcement technique I provide, allows applications to retain meaningful assurances even when taking the necessary action of endorsing user input.

Building Secure Protocols

Without trust between modules, decentralized applications like smart contracts cannot safely rely on many assumptions centralized monolithic systems need for security. I leverage the security notions I develop to construct new secure building blocks needed by practitioners.

I designed and secured the smart contract interface of Town Crier [21], a trusted bridge between blockchain smart contracts and existing web services. Smart contracts provide highly trusted execution, but have no means to query data outside of their blockchain. Town Crier composes a smart contract with trusted hardware, like Intel SGX, to securely connect smart contracts to the outside world, dramatically increasing their potential uses. Smart contract transactions are financially expensive to execute, leading the security of this architecture to rely in part on recouping transaction fees. I thus defined *gas sustainability*, a new safety property that ensures attackers cannot sabotage the service by running it out of money.

I introduced the Public Incompressible Encoding (PIE) [7], a new primitive designed to prove that a server is storing a file with no compression. It operates in the highly-challenging setting where all randomness is public, supporting the needs of decentralized storage startups like Filecoin.[3] Using a construction that is nearly optimal on small files, my results define a useful primitive in the absence of adversarial data, when it composes, but raise feasibility concerns in setting where adversarial data is possible.

I also built Solidus [6], the first publicly verifiable anonymous financial transaction protocol for a *bank-intermediated* system—a structure that conforms to the existing financial system where many users have accounts at a few banks. Solidus transactions are entirely anonymous except to the users and banks involved, while remaining publicly verifiable and auditable. The security of Solidus relies on the Publicly Verifiable Oblivious RAM Machine (PVORM), another new cryptographic primitive. The PVORM is based on Oblivious RAM (ORAM) and is designed to support proofs that data updates are valid according to the protocol without revealing which data is modified.

Cross-Discipline Collaboration

I have also helped colleagues combine security with other disciplines of computer science.

In Obladi [11], I worked with experts on transactional datastores to design, implement, and prove secure an efficient transactional key-value store that obscures all data and access patterns from an untrusted storage layer. I leveraged my familiarity with ORAM and formal notions of privacy to analyze the security of numerous proposed optimizations and provide a proof of security for the final protocol.

[3] `https://filecoin.io/`

I worked with experts in formal logic to develop the Flow-Limited Authorization First Order Logic [14]. FLAFOL combines IFC techniques with formal logic to reason about a concern originally raised by Arden, Liu, and Myers [2]: authorization decisions may cause data leakage if based on secret data. I again focused on security, devising and proving a security property that the combines noninterference of IFC with a conceptually similar, but technically very different, noninterference property of authorization logics [13].

Moving Forward: Practical Compositional Security

Through my research, I have seen the importance of attaining compositional security guarantees, especially for large decentralized systems. A core contribution of my work on reentrancy is the first compositional definition end enforcement technique for secure reentrancy, allowing them to apply to modular applications. My work on PIEs demonstrates how achieving security can be prohibitively expensive in settings where primitives fail to compose.

Existing techniques, however, do not support defining security specifications with compositional properties and realizing them with cryptographic protocols. Cryptographic techniques like UC security say nothing about the meaning of composed specifications, and language-based techniques like IFC generally assume deterministic single-threaded execution. I believe that a combination of these ideas is necessary to provide guarantees to large composed systems like smart contract ecosystems or web pages running JavaScript from many sources. To achieve this combination, I plan to address several important challenges.

Understanding Cryptographic Security Definitions. Cryptographic adversarial models are often described by what an attacker can or will do. For instance, an "honest-but-curious" adversary will follow the protocol as-written, but learn as much as possible. Ideal functionalities define security by actions and responses: "When user U sends request R, perform action A." What happens in a large system when some adversaries are honest-but-curious and others are malicious? Can the combination of two ideal functionalities leak more data than the designer intended?

Recent work by myself and others [1, 8, 9] uses trust relationships to characterize certain attacks, defenses, and adversarial models traditionally defined operationally. In each case, the change of view clarifies how defenses and adversaries may behave as components inside larger systems with more varied trust assumptions. Similarly, ideas like IFC specifications could elucidate how ideal functionalities compose, by tracking what an attacker can influence and what data can leak.

Computational Model of IFC. Real microservice-based applications execute in parallel and rely on cryptography for security, requiring probabilistic analysis that provides security against computationally bounded attackers. Unfortunately, most IFC systems prove security in a much simpler model with deterministic languages and sequential execution. This discrepancy makes it difficult to know what guarantees existing IFC analyses can provide to real systems.

There is very little work into how to express security conditions more complicated than noninterference in a probabilistic setting, and it is not clear how IFC's critical composition guarantees transfer from the deterministic setting to the probabilistic one. Bridging this gap will allow solutions like my IFC-based reentrancy protection to prove the security of, e.g., interacting JavaScript modules relying on cryptography.

Achieving provable compositional security is crucial for securing smart contract ecosystems, large microservice-based systems, and other applications formed by composing modules with complex trust relationships. My work tackles this challenge from multiple angles. I combine techniques from different disciplines to provide new viewpoints on pressing security problems, and actively collaborate with colleagues in other areas to further broaden my perspectives and understand the security implications of other work.

References

[1] C. Acay, R. Recto, J. Gancher, A. C. Myers, and E. Shi. Viaduct: An extensible, optimizing compiler for secure distributed programs. In *42nd ACM SIGPLAN Conference on Programming Language Design and Implementation*

(PLDI '21), June 2021. doi: 10.1145/3453483.3454074.

[2] O. Arden, J. Liu, and A. C. Myers. Flow-limited authorization. In *28th IEEE Computer Security Foundations Symposium (CSF '15)*, July 2015. doi: 10.1109/CSF.2015.42.

[3] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch. iUC: Flexible universal composability made simple. In *25th International Conference on The Theory and Application of Cryptology and Information Security (AsiaCrypt '19)*, Dec. 2019. doi: 10.1007/978-3-030-34618-8_7.

[4] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS '01)*, Oct. 2001. doi: 10.1109/SFCS.2001.959888.

[5] **E. Cecchetti**, A. C. Myers, and O. Arden. Nonmalleable information flow control. In *24th ACM Conference on Computer and Communication Security (CCS '17)*, Oct. 2017. doi: 10.1145/3133956.3134054.

[6] **E. Cecchetti**, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi. Solidus: Confidential distributed ledger transactions via PVORM. In *24th ACM Conference on Computer and Communication Security (CCS '17)*, Oct. 2017. doi: 10.1145/3133956.3134010.

[7] **E. Cecchetti**, B. Fisch, I. Miers, and A. Juels. PIEs: Public incompressible encodings for decentralized storage. In *26th ACM Conference on Computer and Communication Security (CCS '19)*, Nov. 2019. doi: 10.1145/3319535.3354231.

[8] **E. Cecchetti**, S. Yao, H. Ni, and A. C. Myers. Securing smart contracts with information flow. In *3rd International Symposium on Foundations and Applications of Blockchain (FAB '20)*, May 2020. URL `https://ethan.umiacs.io/papers/ifc-contracts-fab20.pdf`.

[9] **E. Cecchetti**, S. Yao, H. Ni, and A. C. Myers. Compositional security for reentrant applications. In *42nd IEEE Symposium on Security and Privacy (Oakland '21)*, May 2021. doi: 10.1109/SP40001.2021.00084.

[10] M. Coblenz, R. Oei, T. Etzel, P. Koronkevich, M. Baker, Y. Bloem, B. A. Myers, J. Sunshine, and J. Aldrich. Obsidian: Typestate and assets for safer blockchain programming. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 42(3), Nov. 2020. doi: 10.1145/3417516.

[11] N. Crooks, M. Burke, **E. Cecchetti**, S. Harel, R. Agarwal, and L. Alvisi. Obladi: Oblivious serializable transactions in the cloud. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, Oct. 2018. URL `https://www.usenix.org/system/files/osdi18-crooks.pdf`.

[12] A. Das, S. Balzer, J. Hoffmann, F. Pfenning, and I. Santurkar. Resource-aware session types for digital contracts. In *34th IEEE Computer Security Foundations Symposium (CSF '21)*, June 2021. doi: 10.1109/CSF51468.2021.00004.

[13] D. Garg and F. Pfenning. Non-interference in constructive authorization logic. In *19th IEEE Computer Security Foundations Workshop (CSFW '06)*, July 2006. doi: 10.1109/CSFW.2006.18.

[14] A. K. Hirsch, P. H. A. de Amorim, **E. Cecchetti**, R. Tate, and O. Arden. First-order logic for flow-limited authorization. In *33rd IEEE Computer Security Foundations Symposium (CSF '20)*, June 2020. doi: 10.1109/csf49147.2020.00017.

[15] P. Li and S. Zdancewic. Downgrading policies and relaxed noninterference. In *32nd ACM SIGPLAN Symposium on Principles of Programming Languages (POPL '05)*, Jan. 2005. doi: 10.1145/1040305.1040319.

[16] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *23rd ACM Conference on Computer and Communication Security (CCS '16)*, Oct. 2016. doi: 10.1145/2976749.2978309.

[17] H. Mantel and D. Sands. Controlled declassification based on intransitive noninterference. In *2nd Asian Symposium on Programming Languages and Systems (APLAS '04)*, Nov. 2004. doi: 10.1007/978-3-540-30477-7_9.

[18] A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification and qualified robustness. *Journal of Computer Security (JCS)*, 14(2):157–196, 2006. doi: 10.3233/JCS-2006-14203.

[19] A. Sabelfeld and A. C. Myers. A model for delimited information release. In *International Symposium on Software Security*, Nov. 2003. doi: 10.1007/978-3-540-37621-7_9.

[20] D. Wikström. Simplified universal composability framework. In *13th IACR Theory of Cryptography Conference (TCC '16)*, Jan. 2016. doi: 10.1007/978-3-662-49096-9_24.

[21] F. Zhang, **E. Cecchetti**, K. Croman, A. Juels, and E. Shi. Town Crier: An authenticated data feed for smart contracts. In *23rd ACM Conference on Computer and Communication Security (CCS '16)*, Oct. 2016. doi: 10.1145/2976749.2978326.