EtherTrust

Endbericht | Call 12 | Projekt ID 2158

Lizenz CC-BY

Inhalt

1 Einleitung

In unserem Projekt EtherTrust beschäftigen wir uns mit der Entwicklung eines Analysetools für Smart Contracts, welches Nutzern und Entwicklern von Smart Contracts mit Hilfe eines Online-Services verfügbar gemacht wird. Das Ziel des Tools ist es, belastbare Garantien für Ethereum Smart Contracts zu liefern, und somit Entwickler beim Erstellen sicherer Verträge zu unterstützen, und Nutzern von Smart Contracts ein Werkzeug an die Hand zu geben um die Vertrauenswürdigkeit von Verträgen besser einschätzen zu können.

Zu diesem Zweck involviert das Projekt auch in einem wesentlichen Maße eine theoretische Modellierung der zugrunde liegenden Analyse und einen Beweis für deren Korrektheit.

2 Projektbeschreibung

Projektziele/ Zielgruppe

Das finale Ziel von EtherTrust ist das Bereitstellen einer Website, auf der Smart Contract Nutzer und Entwickler Verträge auf spezifische Sicherheitseigenschaften hin analysieren können. Diese Analyse soll nicht nur vollständig automatisiert sein (also keinerlei weitere Interaktion des Nutzers benötigen und somit auch kein zusätzliches Wissen des Nutzers über die Analysetechnik voraussetzen), sondern soll auch beweisbare Garantien liefern. Eine Garantie in diesem Zusammenhang ist ein Beweis, dass wenn das Analysetool einen Vertrag als sicher (bezüglich einer bestimmten Sicherheitseigenschaft) einstuft, dieser die Eigenschaft auch zwangsläufig erfüllt. Diese Eigenschaft eines Analysetools wird im Allgemeinen als Soundness bezeichnet. Soundness ist eine Eigenschaft, die vor allem in der Analyse von Smart Contracts eine besondere Relevanz hat: Smart Contracts sind Verträge, die sobald sie einmal in der (Ethereum) Blockchain veröffentlicht sind, nicht mehr verändert werden können. Gleiches gilt für alle Interaktionen mit der Blockchain und somit auch für alle Interaktionen mit den in ihr veröffentlichten Verträgen: Sobald eine Interaktion getätigt ist, kann sie nicht mehr zurückgenommen oder verändert werden. Aus diesem Grund ist es besonders wichtig die Eigenschaften und potenziellen Probleme eines Vertrages zu kennen, bevor man ihn veröffentlicht oder mit ihm interagiert. Entsprechend problematisch sind Analysetools, die Nutzer in falscher Sicherheit wiegen, in dem sie tatsächlich unsichere Verträge als sicher klassifizieren.

Die Zielgruppe von EtherTrust sind somit solche Personen, die sicherheitskritische Smart Contracts entwickeln oder Personen, die mit solchen Verträgen interagieren wollen und dabei

Geld investieren. Von diesen beiden Zielgruppen wird nicht erwartet, dass sie ein tiefgehendes Verständnis der Semantik der Programmiersprache haben, in der die Verträge verfasst werden oder, dass sie formal über die Korrektheit ihrer Verträge argumentieren können. Um trotzdem die allgemeine Sicherheit von Smart Contracts zu erhöhen, bietet EtherTrust die Möglichkeit universelle Sicherheitseigenschaften zu analysieren – also solche Eigenschaften, die von allen vertrauenswürdigen Verträgen erfüllt werden sollen. Dies entbindet die Entwickler und Nutzer von der Bürde, selbst Eigenschaften mathematisch formalisieren zu müssen.

Auch wenn so nicht garantiert werden kann, dass jeder Vertrag exakt das tut, was die Nutzer von ihm erwartet, kann zumindest ausgeschlossen werden, dass er bestimmte problematische Verhaltensweisen zeigt. Dies kann insgesamt die Sicherheit von Ethereum Smart Contracts steigern ohne dabei die Eintrittshürde zum Entwickeln und Nutzen von Smart Contracts zu erhöhen. Im Idealfall führt dies sogar zu einer allgemeinen Steigerung des Vertrauens der Gesellschaft in die Blockchain und Smart Contract Technologie.

Projektergebnis

Das Endergebnis des Projektes ist das Tool EtherTrust, welches Endnutzern über eine öffentliche Website zugänglich gemacht wird. Die Website erlaubt es Nutzern beliebige Smart Contracts in Form von sogenanntem EVM Bytecode (dem Format in welchem Smart Contracts auch auf der Ethereum Blockchain veröffentlicht werden) einzugeben. Diese Verträge werden dann automatisch durch das Tool EtherTrust analysiert: Zu diesem Zweck werden sie in eine logische Repräsentation übersetzt, die eine abstrakte Version der Ausführungssemantik von Ethereum Smart Contracts beschreibt. Diese logische Repräsentation wird dann gemeinsam mit einer logischen Darstellung der zu analysierenden Eigenschaft einem SMT-Solver (einem automatischen Lösungsprogram für bestimmte Formen logischer Formeln) übergeben, welcher überprüft, ob es je passieren kann, dass die enkodierte Eigenschaft verletzt wird. Liefert der SMT-Solver als Ergebnis, dass dies nie der Fall ist, dann ist aufgrund der Art und Weise auf die die Ausführungssemantik und die Eigenschaft abstrahiert und enkodiert wurden garantiert, dass der analysierte Vertrag die entsprechende Eigenschaft erfüllt. Ansonsten besteht die Möglichkeit einer Verletzung der Sicherheitseigenschaft. Dieses Ergebnis wird an den Nutzer zurückgemeldet.

Aktuell unterstützt EtherTrust nur eine generische Sicherheitseigenschaft, die Single-Entrancy genannt wird. Single-Entrancy beschreibt intuitiv die Eigenschaft eines Vertrages (den man sich mehr als Computer-Programm vorstellen muss), im Laufe einer einzelnen Transaktion niemals mehr als einmal ausgeführt werden zu können. Eine Transaktion beschreibt dabei eine Ausführungseinheit auf der Blockchain, die beispielsweise einen Vertrag ausführen kann. Da Verträge nun aber wieder selbst die Ausführung weiterer Subverträge initiieren können, besteht die Möglichkeit, dass ein solcher Subvertrag, den Originalvertrag wiederum als Subvertrag aufruft. Solch ein Verhalten wird im Allgemeinen als gefährlich erachtet, dass Vertragsentwickler diese Möglichkeit häufig nicht in Betracht ziehen und entsprechend kein korrektes Verhalten für

einen solchen Fall implementieren. Das prominenteste Beispiel eines solchen Fehlverhaltens, fand sich im sogenannten DAO-Vertrag, einem Crowd-Funding Vertrag, dem durch einen Programmierfehler Ether im Wert von über 60 Millionen Dollar gestohlen wurde.

Auch wenn wir uns in der aktuellen Version von EtherTrust auf die Single-Entrancy Eigenschaft beschränken, welche eine besondere praktische Relevanz hat, kann EtherTrust sehr leicht um weitere Sicherheitseigenschaften erweitert werden. Dies ist insbesondere dem modularen Aufbau des Tools zu verdanken, welcher sogar eine eigene Spezifikationssprache für abstrakte Ausführungssemantiken und Eigenschaften beinhaltet.

Bei der Entwicklung von EtherTrust haben wir ein besonderes Augenmerk darauf gelegt Infrastrukturen aufzubauen, die leicht von anderen Entwicklern verwendet werden können um das Tool zu erweitern, zu modifizieren und zu verbessern. Nach dem Entwickeln eines ersten Prototyps, der zunächst sehr ad-hoc implementiert war, haben wir das gesamte Tool neu designt und die logische Spezifikation von abstrahierter Semantik und abstrakten Sicherheitseigenschaften vollkommen von der eigentlichen Interaktion mit dem SMT-Solver separiert. Genauer haben wir die Sprache HoRSt entwickelt, welche es erlaubt Menschen lesbare logische Spezifikationen zu schreiben, die dann automatisch (und auf eine optimierte Art und Weise) für den SMT-Solver aufbereitet werden.

Auf diese Weise können Entwickler, die EtherTrust erweitern wollen, die logischen Spezifikationen der Semantik und der Sicherheitseigenschaften leicht abwandeln ohne mit technischen Implementierungsdetails konfrontiert zu werden. Mehr noch, können sie sogar leicht die Semantik für eine komplett andere Sprache implementieren, so dass der Nutzen des Tools nicht alleine auf Ethereum beschränkt ist.

3 Verlauf der Arbeitspakete

3.1 Arbeitspaket 1- *Bootstrapping*

Das Arbeitspaket 1 beschäftigt sich hauptsächlich mit der Literaturrecherche (1.1), der Auswertung existierender Ansätze zur Analyse von Ethereum Smart Contracts (1.2), dem Austausch mit der Community (1.3) und der Formulierung der Problemstellung (1.4). Das Vorgehen entsprach im Wesentlichen der üblichen Vorgehensweise, wie sie in der Wissenschaft praktiziert wird: Wir haben eine ausführliche Recherche existierender wissenschaftlicher Publikationen betrieben und uns auch andere Tools, die nicht in wissenschaftlichen Veröffentlichungen behandelt werden, angeschaut und ihre Funktionsweise analysiert. Des Weiteren haben wir uns intensiv über unsere Erkenntnisse mit anderen Wissenschaftlern (beispielsweise auf Konferenzen) ausgetauscht. Unsere Ergebnisse, eine Übersicht über existierende Ansätze haben wir schließlich in einer wissenschaftlichen Publikation (‚Foundations and Tools for the Static Analysis of Ethereum Smart Contracts', siehe Projektseite), die auch die

allgemeine Problemstellung (nämlich die Eigenschaften, die wir von einem Analysetool für Smart Contracts erwarten) formuliert.

Bei der Bearbeitung dieses Arbeitspaketes haben wir festgestellt, dass leider viele der existierenden wissenschaftlichen Publikationen zu Smart Contract Analysetools, leider weder ihren Code noch genaue Details zur verwendeten Analyse bereitstellen. Dies macht es sehr schwierig, ihre Funktionsweise zu analysieren und mit anderen Tools zu vergleichen.
Ein besonderer Erfolg war es natürlich, dass wir unsere Ergebnisse publizieren konnten und auch ein im Rahmen eines Tutoriums auf einer großen Konferenz präsentieren durften (mehr dazu im Bereich Öffentlichkeitsarbeit).

Planabweichungen gab es im Wesentlichen nicht. Man sollte nur anmerken, dass es sich bei der Recherche in einem so schnelllebigen Bereich wie Kryptowährungen um einen kontinuierlichen Prozess handelt, so dass wir in den anderen Arbeitspaketen auch immer wieder etwas Zeit in die Recherche stecken müssen.

3.2 Arbeitspaket 2 - *Formale Semantik und Sicherheitseigenschaften*

In Arbeitspaket 2 werden die Grundlagen für das Analysetool gelegt, indem zunächst die Semantik von Ethereum Smart Contracts als auch für uns interessante Sicherheitseigenschaften formal definiert werden. Zusätzlich implementieren wir die Semantik auch in einem Beweisassistenten, was uns erlaubt die Semantik zu testen und später auch maschinell verifizierte Beweise zu führen. Die Formalisierung der Semantik war als solche sehr herausfordernd, da die von Ethereum bereitgestellte Beschreibung an vielen Stellen ungenau und fehlerhaft ist und wir so auch konkrete Implementierungen für die Ausführung von Smart Contracts in Betracht ziehen mussten. Um generische Sicherheitseigenschaften zu formulieren, mussten wir viele existierende Probleme von Smart Contracts analysieren um die darunterliegende Problematik zu verstehen und dann formal auszudrücken. Die Ergebnisse, eine vollständige formale Semantik für Ethereum Smart contracts, eine Implementierung im Beweisassistenten F* und formale Sicherheitseigenschaften, die reale Schwachstellen in Smart Contracts ausschließen, haben wir in unserem wissenschaftlichen Papier ‚A Semantic Framework for the Security Analysis of Ethereum smart contracts' (siehe Projektseite) zusammen gefasst, das auf der Konferenz POST akzeptiert wurde. Ein besonderer Erfolg war, dass wir für dieses wissenschaftliche Papier sogar einen Best Paper-Award der Dachkonferenz ETAPS erhalten haben, was eine große wissenschaftliche Auszeichnung ist und zeigt welch großes Interesse an dem Thema unserer Forschung besteht.
Wesentliche Planabweichungen gab es bei der Bearbeitung des Arbeitspaketes nicht. Wir haben jedoch die Beobachtung gemacht, dass die Semantik von Smart Contracts in letzter Zeit häufiger von Ethereum angepasst wurde. Aus diesem Grund, stecken wir aktuell auch Energie darein, die Änderungen einzupflegen und unsere Implementierung zu verbessern, damit diese eine bestmögliche Grundlange für unsere Analyse darstellt.

3.3 Arbeitspaket 3 - *Statische Analyse*

Arbeitspaket 3 widmet sich der formalen Spezifikation der Analyse von Ethereum Smart Contracts. Zu diesem Zweck muss die oben beschriebene Semantik abstrahiert werden und ebenso müssen die Sicherheitseigenschaften abstrahiert werden, so dass sie mithilfe der abstrahierten Semantik ausdrückbar sind. Schließlich soll die definierte Analyse im Rahmen eines Analyse-Tools implementiert werden, so dass Smart Contracts automatisch auf die vordefinierten Sicherheitseigenschaften hin analysiert werden können.

Zu diesem Zweck haben wir die Analyse zunächst theoretisch skizziert und sie dann mithilfe eines Prototyps implementiert um zu evaluieren, ob die theoretische Analyse auch praktisch anwendbar ist. Das Problem liegt darin, dass wir zur automatischen Durchführung der Analyse einen sogenannten SMT-Solver nutzen. Da dieser sehr komplex ist, muss man in einem gewissen Rahmen experimentell evaluieren welche Formulierung und Enkodierung der Analyse zu effizienten Ergebnissen führt. Dies hat sich als komplizierter herausgestellt als wir anfangs dachten, insbesondere auch, weil die Komplexität des Codes des Analysetools über die unterschiedlichen Änderungen der Analyse immer weitergewachsen ist und damit auch immer unübersichtlicher wurde. Obwohl wir zwischenzeitlich einen funktionierenden Prototyp hatten, haben wir uns deshalb entschieden, das Tool noch einmal neu zu implementieren – diesmal auf eine allgemeinere Art und Weise. Das ist uns besonders wichtig, da wir ja am Ende belastbare Sicherheitsgarantien liefern wollen und somit auch eine Implementierung benötigen, der wir das nötige Vertrauen entgegenbringen. Außerdem ist uns bei der ersten Implementierung auch aufgefallen, dass es besserwäre das Tool so zu designen, dass es etwas fortgeschrittenen Nutzern und anderen Wissenschaftlern eine Schnittstelle liefert um neue oder maßgeschneiderte Eigenschaften zu enkodieren. Mit den Erkenntnissen, die wir während des Experimentierens gewonnen haben, konnten wir jetzt ein sehr viel saubereres Tool programmieren, dessen Verbindung zur formalen Analysedefinition klar ersichtlich ist und das leicht angepasst und um neue Eigenschaften erweitert werden kann.

Das Resultat dieser Neu-Implementierung ist jetzt HoRst, eine eigene Spezifikationssprache für abstrahierte Semantiken und Sicherheitseigenschaften. Die Spezifikationen, die in HoRSt geschrieben werden können gleichen mathematischen Spezifikationen (wie sie beispielsweise in EtherTrust: Sound Static Analysis of Ethereum bytecode‘ (siehe Projektseite) gegeben werden). Aus diesem Grund kann eine HoRSt-Spezifikation eine Spezifikation auf Papier ersetzen, bzw. leicht in eine solche überführt werden.

Die Re-implementierung hat auch dazu geführt, dass wir die Erkenntnisse, die wir aus der Phase des Prototyps gewonnen haben, verallgemeinern konnten. In dem wir unterschiedliche Enkodierungen für den SMT-Solver ausprobiert haben, konnten wir nicht nur die Performance unseres Analysetools verbessern, sondern auch im Allgemeinen eine Infrastruktur schaffen, die so viele Enkodierungsdetails wie möglich von dem Entwickler einer abstrahierten Semantik fern hält, so dass sich dieser hauptsächlich auf deren Korrektheit konzentrieren kann.

Da die Re-implementierung und die zuvor diskutierten Optimierungen viel Zeit in Anspruch genommen haben, haben wir uns dazu entschieden lieber mehr Arbeitsaufwand auf diesen Teil zu verwenden und stattdessen, weniger Zeit in die Entwicklung/Implementierung weiterer Sicherheitseigenschaften sowie die Website zu investieren. Unserer Ansicht nach ist es besser für die langfristige Entwicklung des Projektes einen soliden Grundstein zu legen, auf den andere Entwickler leicht aufbauen können anstatt das Tool/die Website direkt um viele, dann entsprechend unausgereiftere, Features zu erweitern.

3.4 Arbeitspaket 4 – *Website*

Das Arbeitspaket 4 beschäftigt sich mit der Entwicklung einer Website als Nutzerschnittstelle für das in Arbeitspaket 4 entwickelte Tool. Ziel des Arbeitspaket war es, eine solche Website zur programmieren, zu testen, und öffentlich zugänglich zu machen. Wie zuvor erklärt, haben wir jedoch mehr Zeit als erwartet auf die Neugestaltung des Analysetools verwendet als anfangs angedacht. Die zusätzlich benötigte Zeit haben wir aus diesem Grund in Arbeitspaket 4 eingespart. Da das Tool erst später als erwartet eine nutzbare Form erreicht hat und diese auch noch keine vollumfängliche Funktionalität beinhaltet (vgl. Diskussion zu den Sicherheitseigenschaften im vorherigen Paragraphen), haben wir zwar eine funktionsfähige Website erstellt, konnten jedoch aus Zeitgründen nicht wie ursprünglich angedacht Community-Feedback einholen und einarbeiten (Tätigkeiten 4.3. und 4.5.).

Um die Website zur Verfügung stellen zu können, mussten wir einen entsprechend leistungsfähigen Server mieten und aufsetzen, auf dem die arbeitsaufwendigen Berechnungen (inklusive des SMT-Solvers) des Tools ausgeführt werden können.

Für die statischen Komponenten der Website haben wir HTML verwendet, eine dynamische Navigation auf der Website konnten wir mithilfe von JavaScript ermöglichen. Um das Tool (welches in Java programmiert ist) in die Website einbinden zu können, haben wir uns das Spring-Framework zunutze gemacht, welches die Entwicklung von Web-Anwendungen basierend auf Java-Programmen unterstützt. Mithilfe dieser Mittel können wir auch ermöglichen, dass die Analyse eines Smart Contracts auf der Website asynchron ausgeführt wird: der Nutzer kann die Website also weiter störungsfrei navigieren, während die Verifikation im Hintergrund stattfindet. Dies ist besonders wichtig, da die Verifikation eines einzelnen Vertrages mehrere Minuten in Anspruch nehmen kann.

4 Liste Projektendergebnisse

Kurzbeschreibung der erreichten Projektendergebnisse jeweils mit Open Source Lizenz und Webadresse (netidee Vorgaben beachten!)

1	***Projektzwischenbericht***	*CC-BY-4.0 International*	*netidee.at/ethertrust*
2	***Projektendbericht***	*CC-BY-4.0 International*	*netidee.at/ethertrust*
3	***Entwickler-DOKUMENTATION*** *des Projektergebnisses für andere Entwickler ("Dritte"), die das Projektergebnis nach Projektende nutzen/weiterentwickeln wollen* *Für Entwickler (Systemkonzept, ggf. Grobspezifikationen):* *a. WAS IST ES* *b. FÜR WEN IST ES /WEM HILFT ES WODURCH* *c. WIE FUNKTIONIERT ES (für Entwickler: Übersicht und detailliertes Systemkonzept, SW-Struktur),*	*CC-BY-3.0 AT*	*netidee.at/ethertrust*
4	***Anwender-DOKUMENTATION*** *des Projektergebnis für Anwender, die das Projektergebnis nach Projektende nutzen wollen* *Für Anwender ("Bedienungsanleitung") :* *a. WAS IST ES* *b. FÜR WEN IST ES /WEM HILFT ES WODURCH* *c. WIE FUNKTIONIERT ES*	*CC-BY-3.0 AT*	*netidee.at/ethertrust*
5	***Veröffentlichungsfähiger Einseiter*** ** Kurzfassung WAS | FÜR WEN | WIE* ** Liste Projektergebnisse -> also diese Liste, ggf. kompromiert* ** mit Angabe Open Source Lizenz/Webadresse* ** wo finden Dritte die Projektergebnisse (inkl. Nutzerdokumentation Anwender bzw. Entwickler)* ** mögliche Weiterentwicklungen/ weitere Einsatz-/ Nutzungsmöglichkeiten*	*CC-BY-3.0 AT*	*netidee.at/ethertrust*
6	***Dokumentation Externkommunikation*** *zur Erreichung Sichtbarkeit /Nachhaltigkeit (eigenes Arbeitspaket vorsehen!)*	*CC-BY-3.0 AT*	*netidee.at/ethertrust*

	** Welche Maßnahmen wurden in welchem Umfang gesetzt* ** Jeweils Bewertung Aufwand / Nutzen* ** Lessons Learned / Empfehlungen für andere Projekte*		
7	***Vollständige EVM Bytecode Semantik*** *(Mathematische Formalisierung der EVM Bytecode Semantik als Small-step-Semantik, die alle Instruktionen umfasst)*	*CC-BY-4.0 International*	*netidee.at/ethertrust*
8	***Implementierung der EVM Bytecode Semantik in F**** *(Implementierung der mathematisch definierten Semantik im Beweisassistenten F*. Diese Formalisierung erlaubt es die Semantik auszuführen und zu testen und kann anderen Forschern als Grundlage dienen um Eigenschaften von EVM Bytecode oder von Analysetools für EVM Bytecode zu beweisen)*	*GNU-GPLv3*	*netidee.at/ethertrust*
9	***Abstrakte Semantik für EVM Bytecode*** *(Semantik für EVM-Bytecode, die die konkrete Small-step Semantik abstrahiert und als Grundlage für eine automatisierte Analyse mithilfe eines SMT-solvers dienen kann)*	*CC-BY-4.0 International*	*netidee.at/ethertrust*
10	***Formalisierung von Sicherheitseigenschaften für smart contracts*** *(Basierend auf der Small-step-Semantik werden formale Eigenschaften definiert, die häufige Sicherheitsprobleme und Bugs in Smart contracts ausschließen)*	*CC-BY-4.0 International*	*netidee.at/ethertrust*
11	***Formalisierung von abstrakten Sicherheitseigenschaften für smart contracts*** *(Basierend auf der abstrakten-Semantik werden die vorher formalisierten Sicherheitseigenschaften approximiert, so dass sie als Basis für eine automatisierte Analyse mithilfe eines SMT-solvers dienen können)*	*CC-BY-4.0 International*	*netidee.at/ethertrust*
12	***Analyse-Framework für Ethereum smart contracts*** *(Framework zum konvertieren von Ethereum smart contracts im Bytecode Format in ein SMT-kompatibles*	*GNU-GPLv3*	*netidee.at/ethertrust*

	Format zusammen mit der Analyse-Logik entsprechend der abstrakten Semantik. Außderdem die Implementierung der abstrahierten Sicherheitseigenschaften in einem kompatiblen Format und die Einbindung eines SMT-solvers zum automatischen Verifizieren der Sicherheitseigenschaften)		
13	***Webservice zum automatisierten Analysieren von Ethereum smart contracts*** *(Website, die Zugang zu dem Analyse-Framework bietet und somit den Nutzern das automatische Verifizieren von smart contracts (bzgl. der implementieren Sicherheitseigenschaften ermöglicht.)*		*netidee.at/ethertrust*

5 Verwertung der Projektergebnisse in der Praxis

Angaben zur Verwertung der Projektergebnisse in der Praxis

Da eines unserer Projektergebnisse eine öffentlich zugängliche Website ist, hoffen wir, dass diese in der Zukunft von vielen Entwicklern und Nutzern von Smart Contracts genutzt wird und somit dazu beiträgt, die Sicherheit von Ethereum Smart Contracts zu verbessern. Da Smart Contracts besonders sicherheitskritisch sind, konnten wir in der Vergangenheit beobachten, dass bereitgestellte Tools zur Verbesserung der Sicherheit von der Community gut angenommen wurden. Unser Tool bietet im Gegensatz zu existierenden Lösungen Sicherheitsgarantien, weshalb wir hoffen, dass viele Nutzer einen Zusatznutzen in seiner Verwendung sehen werden – selbst, wenn sie bereits andere Tools verwenden. Darüber hinaus ist unser Tool durch die bereitgestellte Website leicht nutzbar: es wird keine Installation zusätzlicher Programme benötigt und auch die Rechenlast der Analyse muss nicht durch den Nutzer getragen werden, da alle Berechnungen auf unserem Server ausgeführt werden. Insgesamt hoffen wir deshalb, dass EtherTrust in der Praxis von Ethereum-Nutzern breit genutzt wird und durch die gewährleisteten Sicherheitsgarantien auch dazu beiträgt, dass das allgemeine Vertrauen in Smart Contracts und die darunterliegende Technologie gestärkt wird.

Darüber hinaus hoffen wir durch unsere Arbeit auch eine gute Grundlage für die Weiterentwicklung unseres Analyse-Ansatzes gelegt zu haben: Da wir viel Zeit darein investiert haben, eine eigene Spezifikationssprache zu entwickeln, in der abstrakte Semantiken und Sicherheitseigenschaften ausgedrückt werden können, sind wir optimistisch, dass diese Infrastruktur genutzt wird um das Tool zu erweitern und dabei unser Ansatz, belastbare Sicherheitsgarantien zu liefern, fortgeführt wird. Da unsere Architektur die mathematische Analysespezifikation klar von der Implementierung und Enkodierung trennt, wird das Führen von Korrektheitsbeweisen erleichtert. Weitere Ausführungen dazu finden sich unter Punkt 8.

6 Öffentlichkeitsarbeit/ Vernetzung

Beschreibung der im Rahmen Ihres netidee-Projektes bereits erfolgten bzw. noch geplanten Öffentlichkeitsarbeit oder Vernetzung

Wie bereits im Zwischenbericht beschrieben, haben wir während der Laufzeit des Projektes viel Zeit darein investiert um das Projekt der wissenschaftlichen Community vorzustellen (Präsentationen bei den Konferenzen POST und CAV, sowie einen Vortrag bei der „Summer School on Security & Correctness in the IoT“) und uns mit der Blockchain-Community zu vernetzen.
Seit Abgabe des Zwischenberichtes, haben wir diese Aktivitäten fortgeführt:

Im März dieses Jahres hat Matteo Maffei einen Vortrag zu unserem Projekt beim Helmut Veith Memorial Workshop gehalten. Anfang April hat er unsere Arbeit zudem zunächst bei der „Blockchain Lecture Series in Distributed Ledgers and Smart Contracts“ in Darmstadt präsentiert und im Anschluss ein Tutorial bei der renommierten Dach-Konferenz ETAPS (im Rahmen des „Workshop on Horn Clauses for Verification and Synthesis“) in Prag gegeben. Des Weiteren hat er EtherTrust im Mai beim Workshop zu „Theory and Practice of Blockchains“ in Aarhus und im August bei der FOSAD Summerschool in Bertinoro (Italien) vorgestellt. Zusätzlich hat Clara Schneidewind das Projekt im Februar 2019 bei ihrem Forschungsaufenthalt an der University of Pennsylvania präsentiert.

Wir hoffen, dass sich mit Fertigstellung der Website nun weitere Möglichkeiten ergeben das Projekt mehr praktischen Anwendern bekannt zu machen. Insbesondere planen wir die Website über unsere Social-Media-Kanäle zu bewerben. Zudem wollen wir in Kürze ein weiteres wissenschaftliches Papier bei einer internationalen Konferenz veröffentlichen, das EtherTrust und die Spezifikationssprache HoRSt vorstellt. Wir hoffen damit weitere Bekanntheit in der wissenschaftlichen Community zu erlangen und somit andere Wissenschaftler zu animieren auf unserem Tool aufzubauen.

7 Geplante Aktivitäten nach netidee-Projektende

Sind weiterführende Aktivitäten nach dem netidee-Projektende geplant?

Da wir denken, dass das Projekt weiterhin großes Potential hat, wollen wir auch nach Projektende aktiv daran weiterarbeiten. Auf jeden Fall werden wir die Website, für die wir nun bereits alle Infrastruktur geschaffen haben, aufrechterhalten und mit Erweiterung des Tools auch um entsprechende Features an der Nutzerschnittstelle erweitern. Um das Tool zu erweitern wollen wir mit unterschiedlichen Formulierungen der abstrahierten Ausführungssemantik experimentieren um sowohl die Laufzeit des Tools als auch seine Präzision zu verbessern. Des

weiteren wollen auch neue generische Sicherheitseigenschaften formulieren, auf die hin Verträge dann automatisch überprüft werden können. Im Rahmen dieser Erweiterungen planen wir auch die Ausdrucksstärke unserer Spezifikationssprache HoRSt zu verbessern und weitere Optimierungen in der automatischen Übersetzung von HoRSt in das Enkodierungsformat des SMT-Solvers vorzunehmen. Dies würde auch anderen Entwicklern, die EtherTrust weiterentwickeln oder modifizieren wollen zugutekommen.

8 Anregungen für Weiterentwicklungen durch Dritte

Welche Nutzungs- und Weiterentwicklungsmöglichkeiten für Dritte ergeben sich durch Ihr netidee-Projekt bzw. empfehlen Sie?

Eine direkte Nutzungsmöglichkeit ergibt sich natürlich direkt aus dem bereit gestellten Tool und der Website, die einen leichten Zugang dazu ermöglicht. Wir hoffen, dass EtherTrust (eventuell in Konjunktion mit anderen Tools) in den Entwicklungsprozess sicherheitskritischer Smart Contracts integriert wird.

Darüber hinaus sehen wir großes Potential in der Weiterentwicklung von EtherTrust durch andere Wissenschaftler oder Designer von Analysetools: Wie bereits in den vorherigen Punkten des Berichtes angesprochen, haben wir uns bemüht EtherTrust möglichst modular zu gestalten, so dass andere Entwickler und Wissenschaftler auf unsere Arbeit aufbauen können. Der größte Modularisierungsschritt dabei ist die Entwicklung der Spezifikationssprache HoRSt, die es erlaubt logische Analysespezifikationen auf eine übersichtliche und menschenlesbare Art und Weise zu schreiben. Spezifikationen, die in HoRSt geschrieben sind, werden dann in ein Format übersetzt, welches als Eingabe für SMT-Solver dient. Dabei werden automatisch Optimierungen durchgeführt, die zu einer (für den SMT-Solver) möglichst performanten Enkodierung führen. Folglich müssen Entwickler, die die abstrahierte Ausführungssemantik, die wir für EtherTrust erweitern oder abändern wollen, lediglich die menschenlesbare HoRSt-Spezifikation verändern. Gleiches gilt für die Veränderung bzw. die Erweiterung um Sicherheitseigenschaften: Auch diese werden in HoRSt enkodiert und sind aus diesem Grund leicht leserlich und modifizierbar. Zusätzlich dazu kann HoRSt auch verwendet werden um Programanalysen für andere Programme als Ethereum Smart Contracts zu entwickeln.