SmartPiggies

An open-source standard for a free peer-to-peer global derivatives market

Version 0.3.1 March 7, 2019 (Initial Draft: November 12, 2018)

A current version of this paper is available at www.smartpiggies.com

Michael Arief
Toby Jaguar Algya
Alexander Lee

Contact: realsmartpiggies@gmail.com

Abstract

This document presents a theoretical description of an open-source standard for a decentralized free peer-to-peer global derivatives market. This standard is intended to be compatible with open decentralized networks which incorporate smart contracts that are able to support all required functionality. The standard outlines an instrument that provides a risk management mechanism similar to that of traditional financial options while leveraging the inherent advantages of smart contract design. SmartPiggies are a novel collateral-backed non-fungible token created via a transaction with a smart contract, for which the authors provide an interface specification in the Solidity programming language. The interface defines facilities for purchase and sale of these tokens without the need to rely upon external market mechanisms. SmartPiggies tokens are fully transferable between accounts, and can be cleared and settled directly on a decentralized network. This document also explores the potential risk factors users will face when dealing with SmartPiggies as well as design decisions made to minimize those risks. The ideas in this paper are presented with the public benefit in mind, and the design presented does not seek rent nor revenue for the authors.

Table of Contents

Introduction

The idea of cryptographically secured financial assets has expanded and evolved rapidly over the last decade. Since Satoshi Nakamoto's publication of the Bitcoin whitepaper in 2008, the idea of decentralized, digital cash has spread far beyond its roots as a theoretical topic on a cryptography mailing list, inspiring broad interest from entrepreneurs, investors, and the general public. This interest has stimulated the flow of intellectual capital towards the cryptocurrency space and has has ushered in a number of startups, investment funds, and research initiatives, along with numerous copycats and competitor blockchains. This movement has also spun off separate technology projects inspired by the core idea of a trustless decentralized network for computation and accounting.

The Ethereum network is one such project which expanded the facility of decentralized blockchain-based systems to include smart contracts capable of processing arbitrary computations.[1] Following the publication of the Ethereum whitepaper and subsequent launch of the network, numerous startups were created to develop applications and businesses on top of Ethereum. Many technologies are now available on the Ethereum platform, including the digital equivalents of U.S. dollars and other commonly used currencies. With these new technologies, novel financial products denominated in “real-world” units of account may be constructed which offer users utility well beyond that of digital cash payments alone. Inspired by ERC-721 non-fungible tokens (NFTs), ERC-20 stable tokens, decentralized oracles, and the Ethereum network's flexible smart contracts, this paper outlines a theoretical description and open-source standard for options contracts issued on the Ethereum network which can be used to hedge financial risks priced in any supported currency.[2] The authors term these contracts SmartPiggies tokens, as they are akin to piggy banks (collateralized options) which may be "broken open" under certain conditions, allowing the counterparties to claim portions of the stable token collateral stored inside.

The Case for Decentralized Options

A primary benefit of decentralized options is that they enable anyone who can access the appropriate public decentralized network to use sophisticated financial hedging instruments. By “decentralized option,” the authors mean an option (at the highest level of abstraction, not a legal contract per se) which exists on a decentralized network.[3] Retail options markets may not

[1] While the Ethereum smart contract virtual machine assembly language is often touted as being “Turing-complete,” due to certain deliberate technical limitations all programs will eventually halt, so “arbitrary” is used somewhat loosely here.

[2] The authors assume that the reader has an understanding of traditional financial options throughout this paper. For a primer on traditional options, refer to Appendix I.

[3] Much has been written on the nature and benefits of decentralized networks such as Bitcoin and Ethereum; for purposes of this discussion the authors assume that the reader is familiar with several of these ideas. See, for example, Antonopolous 2016, “The Internet of Money” for an overview of these benefits.

be available everywhere despite demand for their services, and even in jurisdictions where retail options markets do exist (such as the United States), most customers are unlikely to have access to over-the-counter (OTC) markets where they can purchase more sophisticated protection tailored to their individual needs. Decentralized options with flexible parameterization allow anyone with access to the network to create, sell, and purchase options with any parameters and reference to any underlying instrument for which a price can be accessed.

Counterparties using decentralized options can bypass legacy intermediaries and directly engage in options writing and trading activities with lower friction and at lower cost. Furthermore, because automated marketing mechanisms can be built directly into these smart assets, the network they run on can itself behave as a globally unified marketplace. This is especially important for bespoke, illiquid assets, as market makers will be able to provide advanced financial services and risk management tools to customers across the globe, who, due to economic, regulatory, or other constraints, are not currently served by any party in this capacity.

In addition to enabling access to a broader base of potential customers, decentralized options as described also offer users a number of cryptographic assurances. A decentralized option may be transparently collateralized such that the collateral is locked up prior to exercise and is guaranteed to be available to the holder should it be exercised in the money, eliminating significant counterparty risk.[4] Additionally, if using a stable token for the collateral, the purchaser of a decentralized option can also limit exchange rate risk by utilizing a stable token which matches the currency of the risk exposure they are hedging with the option.[5]

Another benefit of decentralized options is the possibility of sovereign ownership of the asset. Rather than relying on a broker to custody and settle the derivative, which exposes holders of options to a form of counterparty risk, the holder can directly own and control the option via their private keys on a decentralized network. They may additionally take advantage of multisignature schemes which allow for more sophisticated and flexible forms of ownership and control than a traditional online brokerage may offer.

Decentralized options offer the potential to protect against changes in the price of a broad spectrum of assets, well beyond the scope of traditional options offered to retail investors today. Using these options, a pig farmer could protect against movement in the prices of gasoline, livestock feed, or agricultural output, a small import-export business owner could hedge against fluctuations in exchange rates for local currency, and a cryptocurrency trader could insure the value of a portfolio. For any asset, product, or service with a suitable price feed, a decentralized

[4] The authors expect that any user of a decentralized option would prefer an open source implementation to allow vetting and verification of the logic that will be used to execute the options.
[5] This latter risk limitation is not, however, cryptographically guaranteed; see section on “External Contract Interaction” for further detail on stable token risks.

option offers protection against unexpected fluctuations in that price, enabling individuals to manage risk in their personal finances and business ventures.

SmartPiggies: Decentralized Options

The SmartPiggies design enables the creation of non-fungible tokens on Ethereum which function similarly to traditional financial options, with novel extensions made possible by smart contract design.[6] These NFTs are collateralized with ERC-20 tokens and utilize oracles to fetch settlement prices, with the pre-committed collateral rendering SmartPiggies similar to “capped” or “barrier” options.[7] SmartPiggies tokens may be transferred between accounts like other tokens on Ethereum, and have innate mechanisms for autonomous sale and purchase. The subsequent sections will lay out the overarching design of the SmartPiggies system, and walk through an example of their creation and use.

Core Components of the SmartPiggies Design

The SmartPiggies design includes a number of core components which enable the creation of a decentralized option, namely non-fungible tokens (NFTs), oracles, and stable tokens. Each of these components described below should be understood individually in order to understand the design of the overall system. If the reader is already familiar with some or all of these components, they may skip the corresponding sections without loss of context.

Non-Fungible Tokens

Non-fungible tokens (NFTs) are digital tokens which can have individualized characteristics per token.[8] Non-fungibility means that the tokens are not of the same “intrinsic” value as one another due to their distinct properties, and that a holder of an NFT would not (necessarily) freely exchange it for a different NFT issued by the same contract. This is in contrast to fungible tokens such as those specified in the Ethereum ERC-20 standard -- any ERC-20 token issued by a particular contract is theoretically identical in value to any other issued by the same contract. Note that while NFTs are not fungible, they are as easily transferable as other tokens on Ethereum such as ERC-20s.

---

[6] Though SmartPiggies could be built atop any decentralized network supporting the required elements of the system, the design will be outlined using Ethereum, which the authors utilized to build a prototype implementation.

[7] A true barrier option, unlike a SmartPiggies NFT, will automatically be settled if the price of the underlying passes a “barrier” specified at the time the option is written; the limited collateral of a SmartPiggies NFT functions similarly to a barrier but the token will not automatically be settled if the holder would be able to realize the full collateral by exercising the token. Such a programmatically-settled barrier option could potentially be constructed based on the SmartPiggies design with further reliance on external data feeds, but is not included in the core SmartPiggies design.

[8] The ERC-721 standard is a well-known specification for NFTs on Ethereum which served as a point of inspiration for the SmartPiggies design; see http://erc721.org/ for details.

Given that SmartPiggies options could have all manner of different values for their core parameters, a non-fungible token specification is a natural base for the standard. Inspired by the ERC-721 design, the authors have outlined a novel NFT specification with particular considerations made for decentralized options, intended for use with SmartPiggies.

Oracles

An oracle is an agent which acts as a bridge between a decentralized network such as Bitcoin or Ethereum and data external to that network. Intrinsically, these networks are encapsulated systems, limited to the data which has been recorded on their respective blockchains. Without external data being provided to the chain, decentralized applications are limited to interacting with data about “on-chain” transactions.[9] There is the potential for a great deal of added value if decentralized applications are able to additionally access data which is generated “off-chain,” and oracles work to provide this access.

Oracle services are provided by a number of different entities, which take different approaches to the problem of bringing off-chain data to the network.[10] While exact implementations differ, the broad design typically involves an on-chain smart contract monitoring requests for external data, and an off-chain system to handle those requests, fetch the data, and report it on-chain via the oracle smart contract. This introduces issues of trust around the source and provision of the data as they are not intrinsically verifiable on-chain. Further discussion of this important topic follows in the “External Data Risks” section below.

By design, the SmartPiggies smart contract interfaces with proxy smart contracts rather than directly communicating with a specific oracle service to resolve price data. These proxies (or “resolvers”) have a standard interface for the SmartPiggies smart contract and can be specifically designed to relay data from any given oracle service. In this way, the SmartPiggies design is free to interface with any external data provider that can be accessed via proxy, including new service providers which may arise in the future.[11]

As SmartPiggies tokens aim to offer hedges against financial assets outside of the Ethereum network, the use of oracles to fetch external market price data is currently a necessity for settlement of options with external underlyings. If at some point in the future the relevant data providers (e.g. stock exchanges, Bloomberg, etc.) provided signed data directly on-chain via their own smart contracts, the use of oracles and some of the risks associated with them could be avoided.

---

[9] “On-chain” transactions refer to those executed entirely on the decentralized network by the underlying protocol or scripting language, such as Bitcoin Script or the Ethereum Virtual Machine (EVM).
[10] Two candidate oracle providers for SmartPiggies, Oraclize and Chainlink, are discussed in greater detail in the section “Oracles as a Bridge to the Blockchain” below.
[11] Existing oracle services already use different request formats to query the same API, and the authors are not aware of any accepted standard for querying an external API endpoint as of this writing.

Stable Tokens

The third core component of the SmartPiggies design is a stable token to use as collateral. The ERC-20 specification provides for fungible, transferable tokens on the Ethereum network, and a number of ERC-20 stable tokens have been created which specifically aim to maintain stable parity with a particular reference, often a fiat currency such as USD. Different stable tokens take different approaches to maintaining their peg (often 1:1 between the token and base unit of the reference), even for the same reference. Some risks associated with each of these approaches are discussed in the “Stable Token Risks” section below.

Stable tokens allow for SmartPiggies tokens to be used as hedges against financial assets outside of Ethereum priced in the same currency to which the stable token is pegged. This is crucial for enabling the use of SmartPiggies tokens for risk management purposes, as the currencies of the hedge and the underlying need to match to allow for direct protection against movement in the underlying. For example, if a user of SmartPiggies has exposure to a particular stock priced in USD, and wishes to hedge her exposure to that stock, she may do so by using a SmartPiggies token that would pay out in a stable token pegged to USD, offsetting USD movement in the price of the stock.

By using stable tokens and oracles in concert, non-fungible SmartPiggies tokens allow for options on assets outside of a decentralized network to be created and traded directly on the network, hedging risks in the same currency as those assets. The following sections describe a design for SmartPiggies implemented on top of Ethereum, which currently supports all of the required functionality.

Elements of a SmartPiggies Token

The core features of a traditional option are all part of the SmartPiggies token design: there is a reference to the underlying, a strike price, an expiry, and a lot size, in addition to identifiers for the style and direction of the option. The design also includes a number of extensions to support the sale and purchase of the tokens as well as on-chain clearing and settlement.

A key extension enabling settlement of SmartPiggies tokens is a reference to an intermediary contract with standardized I/O,[12] which will communicate with an oracle or other data provider to fetch the price used in calculation of the settlement. Equally important is a reference to the ERC-20 stable token contract which will be used by the SmartPiggies smart contract to manage payment of the collateral to the appropriate parties at settlement.[13] The SmartPiggies smart contract also defines variables to capture the writer and holder of the option, which are used for transfer and settlement of the tokens.

[12] Here, defining a standardized input/output interface.
[13] If using the autonomous auction capability of the token, an ERC-20 contract is also specified for managing payment of the premium.

Beyond these external smart contract and counterparty references, the SmartPiggies design also incorporates variables used to manage the native sale and purchase of SmartPiggies tokens (that is, without using a third-party smart contract to facilitate exchange). Key among these is an identifier defining the SmartPiggies token as either an option (the case which has been referred to thus far), or as a “request” for an option (termed RFP, or “request for piggy”). The terms “SmartPiggies option” and “SmartPiggies RFP” may be used in the remainder of this paper to refer to a SmartPiggies token in one of these two states. The RFP case is somewhat complex, and is detailed below in the section on SmartPiggies functionality. For now, it is sufficient to understand that an identifier exists which defines the SmartPiggies token as an option or an RFP.

In addition to the identifier described above, a number of parameters related to the auctions which can be used to either sell a SmartPiggies option or find a fulfiller of a SmartPiggies RFP are specified. These include the auction start price, reserve price, time step (in blocks), price step, auction expiry (a block height), and a boolean variable indicating whether or not an auction is active. Autonomous auction mechanisms comprise the core extensions of the SmartPiggies standard beyond traditional options. In practice, additional variables will be required to implement the NFT specification, but we leave these details aside for now as they are not requisite to understanding the functionality of SmartPiggies.

SmartPiggies Functionality

SmartPiggies functionality is relatively complex due to intricacies in the process of on-chain settlement of options with an off-chain underlying. Complexity also arises from extensions of the SmartPiggies token which offer possibilities not available with traditional options. Despite this complexity, the functionality can be encapsulated into a number of categories which shall be addressed individually below.

Reclamation of Collateral

In the case that the current holder of a SmartPiggies token is also the counterparty who collateralized the option as the writer, they may opt to “break open” the token and reclaim the collateral associated with it (as delegated to the SmartPiggies smart contract via the reference ERC-20 stable token contract). Users may wish to do this for a number of reasons: for example, if the original token creator accidentally specified illogical or incorrect parameters during creation of the token (such as an expiration millions of years into the future, or a non-existent underlying reference for the oracle to query), she may wish to essentially “re-make” the token with corrected parameters. Additionally, if a writer specified a particular combination of underlying, strike, and collateral that seemed economically reasonable at one point, but due to unforeseen movement in the price of the underlying now seems very unfavorable, she may wish to remake the token with a change to, for instance, the strike parameter (assuming the token has not yet been sold). The functionality to essentially “do-over” the creation of a SmartPiggies

token is fairly straightforward and is primarily provided as a convenience for correcting errors made during the process.

As a secondary use case, this functionality may be leveraged for third-party resolution or court adjudication in the case of a dispute over settlement due to a failure of the assigned data resolver. Both counterparties would have to submit themselves to a third-party arbitrator such as a managed smart contract or a court in a jurisdiction to which they were subject. The SmartPiggies token writer would be asked to submit a sufficient amount of collateral that the arbitrator would hold in escrow, and the token holder would be asked to transfer the SmartPiggies token to the writer (possibly via the third party) who could then reclaim the collateral. Once the original token writer had taken possession of the original collateral, the arbitrator would be in a position to arrive at some equitable resolution at its leisure.

Native Sale

SmartPiggies tokens by design have a capability which enables the current holder to allow the token to effectively market itself. Specifically, the native sale mechanism for SmartPiggies is a form of Dutch auction, whereby the holder of a token may elect to define a starting price, minimum reserve price, price step, and time step, subsequently marking the piggy as “for sale.” When this auction mechanism is activated, the token will hold a Dutch auction for itself: the initial price will be the starting price specified by the holder, which will decrement along the slope defined by the ratio of the price step to the time step (specified in blocks) until it meets the reserve price. The auction may only be started if the auction parameters are set, and may be ended at any time by the current token holder as long as a a transaction to purchase the token is not currently taking place.

A buyer scanning the Ethereum network will be able to find any SmartPiggy options marked as “for sale,” and may invoke the appropriate function to purchase the token.[14] The current design of the SmartPiggies token specification requires that payment be in the form of an ERC-20 token delegated to the SmartPiggies smart contract (which need not be the same ERC-20 used for collateral); an alternative design not adhering to the specification could allow for payment directly in Ether.[15] Regardless of payment implementation, the auction would not be able to be cancelled during the execution of the purchase function, and would revert if a transaction with insufficient payment were sent to it. Upon successful execution of the purchase function, all state variables related to the SmartPiggies token’s ability to auction itself would be reset, and the address which successfully called the purchase function would be recorded as the token holder. The new holder would then be free to set the token up to auction itself again with a newly chosen set of parameters.

---

[14] The current price for a SmartPiggies token using the autonomous auction facility is easily calculable from the public auction parameters and the current block height: $p_t = p_0 - ((b_t - b_0) / s_b * s_p)$ where p_t is the current price to secure the token, p_0 is the auction start price, b_t is the current block height, b_0 is the token creation block height, s_b is the block step size, and s_p is the price step size.

[15] The authors deliberately chose to disallow payments in Ether due to security considerations; see subsection on “Honeypot Risk” in the “Risk and Mitigations” section for further detail.

An optional whitelist may also be specified when setting up a SmartPiggies token for native sale, which could be limited to a single address.[16] This would allow the holder to limit their sale of the token to pre-screened buyers or a single counterparty with whom they have previously negotiated the terms of the option. Defining a limited whitelist enables deals similar to those in OTC options markets today, with the added security guarantees provided by the Ethereum network. A similar structure could additionally be implemented as a blacklist, if sellers wished to ensure that their transaction was in compliance with, for example, regulatory sanctions against a known list of addresses. While the authors are aware that blacklists may be ineffective in the context of open decentralized networks, this may be necessary for users in certain jurisdictions.

Using a Dutch auction for the native sale mechanism strikes what the authors believe to be a convenient balance between flexibility in pricing for the seller (potentially allowing sellers to capture a higher premium than their reserve price when demand is high), and the creation of effective “market depth.” This balance is met by allowing SmartPiggies tokens to exist as available-for-sale in a long enough time window for potential buyers to browse and purchase them, all without the need for any kind of traditional exchange infrastructure to exist. Dutch auctions also flexibly allow for single-price sale as a special case, where the starting price and reserve price are the same. Different implementations of the SmartPiggies design could, of course, elect to include a different kind of native sale mechanism as desired.

It is important to note that use of the native sale mechanism is entirely optional: regardless of implementation, holders of SmartPiggies tokens could simply use third-party mechanisms such as escrowed auction smart contracts or exchanges instead, according to their preference. The authors include the native sale mechanism to allow for immediate marketability of SmartPiggies tokens without the need to rely on external auction contracts, decentralized exchanges, or exchange listings.

Native Purchase

As noted above, when a SmartPiggies token is created, it may be instantiated in one of two states: as an option or an RFP. In the default case, the SmartPiggies token will be created as a collateralized option, with the expected option parameters (underlying, strike, expiry, etc.) In the alternative case of an RFP, the SmartPiggies token will be initially created as a collateralized request for an option with those same parameters.[17]

When a SmartPiggies token is in the RFP state, certain functionality is temporarily unavailable (namely, the ability to put the token up for sale, and functions relating to calling the oracle

[16] In Solidity, this would require specification of an array field even if the whitelist contained only a single address, which would increase the cost of marketing the token.
[17] The collateralization in this case refers to collateral put up by the requestor for the premium to be paid for the requested option. The “collateral” parameter of the RFP would be a desired amount of collateral that the creator wished to have attached to the option.

service and settling the option, as the option technically does not yet exist). In place of these, functionality is enabled which allows the requestor to natively find a potential provider of a SmartPiggies token matching the option parameters specified (that is, without needing to rely on external smart contracts). We term this the “native purchase” capability of a SmartPiggies token in the RFP state.

This native purchase capability is provided by two functions in the contract. The first function is the same auction function used for native sale, with the same parameters: starting price, reserve price, time step, price step, auction expiry, and a boolean indicating whether or not the auction is active. The difference is in the nature of the auction when activated for a token in the RFP state: whereas the native sale mechanism uses a Dutch auction format, the native purchase mechanism is a reverse Dutch auction. The requestor will specify a low starting “bid” for the premium that she would pay in exchange for the particular SmartPiggies option as defined by variables noted above, along with the extra parameter for RFPs denoting the desired collateral for the option itself. At each time step while the auction is active, the offered premium will increase by the price step until the reserve price is met and/or the auction expires.

Prior to starting the auction, the requestor must have delegated at least the reserve price in the reference premium ERC-20 token to the SmartPiggies smart contract, which takes ownership of that reserve amount for the duration of the auction.[18] Similar to the Dutch auction used in the native sale mechanism, the entire price schedule will be known in advance, allowing potential fulfillers of the RFP to know what premium they would have to be willing to accept in exchange for collateralizing the requested SmartPiggies token at any particular block height. Furthermore, given the collateralization of the reserve price which is now held by the SmartPiggies smart contract, any potential fulfillers can be assured that they will in fact receive the premium should they wish to write the option. Given the flexible parameterization, the requestor may of course also set the reserve price equal to the starting price if she wishes to specify a fixed premium instead of using the full auction functionality.

The second piece of the native purchase capability is a function which allows a willing provider of collateral to receive a premium in exchange for collateralizing the requested option.[19] Mechanically, this works as follows: the entity invoking the function to satisfy the auction must first delegate at least the amount of requested ERC-20 collateral to the SmartPiggies smart contract. When the function to end the auction is invoked, the SmartPiggies smart contract takes ownership of that option collateral, and in exchange transfers the premium payment in ERC-20 tokens to the address satisfying the auction.[20] With the two-way transfer of ERC-20 token ownership complete, the collateral provider is recorded in the SmartPiggies smart contract

[18] Recall that this need not be the same ERC-20 as specified for the option collateral.
[19] This function may only be invoked for a SmartPiggies token which is in the RFP state and actively being auctioned.
[20] If the reserve price for the auction has not yet been met, any difference between the reserve price and premium paid is returned to the requestor, as the reserve price was required to be fully collateralized at the start of the auction.

as the option writer, the original creator of the RFP remains the holder, and the state of the SmartPiggies token changes from RFP to option.[21] The net result of this operation is that the original requestor now has a fully collateralized SmartPiggies option instead of an RFP, and the provider of that option has received a premium in return -- all accomplished on-chain without a third-party exchange smart contract.

Collateral Splitting

If the holder of a particular SmartPiggies token wishes to split the collateral into multiple NFTs, she may do so as long as it is in the option state and is not currently being auctioned.[22] To split the token, the holder invokes a function specifying a single integer which must be less than the amount of collateral held by the SmartPiggies smart contract associated with the token being split. The splitting function executes the creation function from the SmartPiggies smart contract twice, with all parameters the same as those of the token being split except for the collateral amount: the integer passed to the function is used for one collateral amount, and the collateral associated with the pre-split token minus that integer is used for the second amount. At the end of the splitting function, the original token is burned.

After calling the split function, the holder now has two newly minted SmartPiggies tokens. These two tokens share the same state parameters which define them as an option, and have collateral amounts summing to the original collateral amount of the split token. The SmartPiggies smart contract retains control of the collateral throughout this process.

Settlement Functions

The settlement process of the SmartPiggies token design encompases a series of steps to exercise the option. If the holder of the SmartPiggies token wishes to exercise the option (at any point if it is an American option, or after expiry if European), she may initiate the settlement process, which includes resolving the price of the underlying, calculating collateral distribution (i.e. determining the payout owed to the holder), and disbursing collateral payouts.

To begin the settlement process, the SmartPiggies smart contract calls an oracle (via a standardized proxy) to resolve the price of the underlying. The SmartPiggies token holder executes a clearing function that requests a predefined oracle smart contract to retrieve a price for the underlying specified at token creation. The oracle smart contract will signal to an oracle server that a price has been requested.[23] The oracle server will subsequently retrieve the requested price from an API endpoint and supply that price back to the oracle smart contract, which will in turn relay the price to the SmartPiggies smart contract to be recorded. Once this

[21] By design, successful execution of the native purchase function is the only mechanism for changing the token state in this manner, and fulfillment of an RFP may happen at most once during the lifetime of a SmartPiggies token.
[22] The holder may wish to split the token as part of a risk rebalancing scheme, or if the collateral size were inconvenient to market, for example.
[23] This is an off-chain process. Please see the section on “Mechanisms for Handling Off-Chain Data.”

process of fetching the settlement price is complete, the actual settlement calculation can be carried out.

Either counterparty may execute the function to calculate collateral distribution rights once the settlement price has been saved for a particular NFT. In this step the smart contract logic calculates any payout rights to the counterparties, which will be delegated to them from the ERC-20 collateral associated with the SmartPiggies token. Depending on the settlement price and the option parameters, one counterparty may be delegated all of the collateral, or it may be split between the two counterparties. This process is only a transfer of rights to payments; the next and final step involves the actual transfer of collateral itself from the SmartPiggies smart contract to the counterparties.[24]

In order to claim collateral, the counterparties who have been granted rights to a share of the collateral in the previous step may invoke a separate function which will transfer a specified amount of ERC-20 collateral tokens they are due. This payment is structured as a separate "pull" payment rather than a "push" payment combined with the prior step due to security and accounting concerns.

There are many permutations of this settlement process, as users may be party to different SmartPiggies options tokens, e.g. American style vs. European, put vs. call, etc. However, upon exercise of a SmartPiggies option, this clearing, settlement, and withdrawal process is the same. Rather than withdrawing all of the collateral at once, a counterparty may also, elect to make partial withdrawal of funds for convenience, accounting, or execution cost considerations.

Contingency Settlement Arbitration

As no data source can be guaranteed to be correct or incorruptible, there may exist a desire to introduce a trusted third party arbitrator into the settlement process of a SmartPiggies option, particularly if that option carried a large amount of collateral. The arbitrator, along with the holder and writer, would hold one key for a 2-of-3 multisignature emergency intervention, and would therefore have to be considered fair and neutral by both counterparties. The writer would choose the arbitrator upon creation of the SmartPiggies token, the arbitrator would have to acknowledge the assignment, and the holder would be aware of the arbitrator prior to purchase. If this emergency intervention option is enabled, the recorded settlement price (or lack thereof due to an oracle failure) would be subject to review and potential amendment for a period of time before either counterparty could trigger calculation of the settlement and claim of the collateral. An emergency intervention would require the aggrieved party to seek out the assistance of the arbitrator to execute the 2-of-3 multisignature scheme to correct a faulty settlement price within the allotted intervention period.

[24] After this step has been completed, the original token may be burned as ERC-20 collateral balances owed to various accounts are held in a separate data structure.

The authors imagine a network of reputable arbitrators naturally forming. Reputable arbitrators would add value to SmartPiggy tokens as purchasers would have additional assurance that settlements would conclude fairly. Reputable arbitrators would be compensated for the value they add by the writers and low reputation or disreputable arbitrators would not add much value, or perhaps negative value.

Two-Party Reassignment of External Data Source

In the event that an external data source goes defunct at any point during the life of a SmartPiggies token, both counterparties may agree to change the data source used to settle the option. One counterparty may propose a change along with an incentive payment that the other party may or may not accept. The proposal may be revoked at any time.

Two-Party Escape from the Prisoner's Dilemma

If a SmartPiggies token appears to be unresolvable, the counterparties may agree to initiate a game to resolve the option. This may be desirable if the external data source is not available after the expiration of a SmartPiggies token and no trusted third party arbitrator has been assigned (or has declined to intervene), and if both parties cannot come into agreement to reassign the data source or a trustworthy source is no longer available.

In this game, both counterparties may make an offer to claim a percentage of collateral. Either counterparty may accept the offer of the other, with the remaining percentage implicitly assigned to the accepting party. As an economically rational actor should have a preference for some fractional recovery over a complete loss, this mechanism should be capable of serving as a final fallback for resolution of a token in the absence of a functioning, mutually agreeable data source.[25] Furthermore, if the underlying prices were generally known to both counterparties as would be expected, the "last known potential settlement price" would function as a Schelling point to negotiate around via this game mechanism.

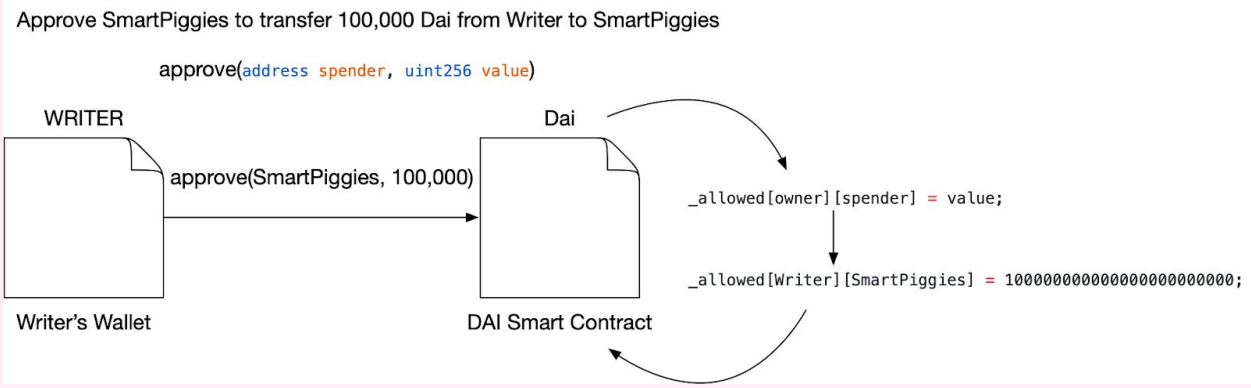
SmartPiggies Token Lifecycle: An Illustrated Walkthrough

The following describes the core functionality of SmartPiggies with a worked example, in which a writer creates a SmartPiggies token representing an American put option on the February 2020 Lean Hog Futures contract on the Chicago Mercantile Exchange (CME:HE.G20) to be cash settled in Dai. The purchaser of this token would be protected against near-term declines in the expectation of future prices for lean hogs.

[25] By "economically rational actor," the authors intend the theoretical model of such an actor, which may not exist in the real world. The model is nevertheless useful for making certain game-theoretic arguments; in this case the argument is that two players in a prisoner's dilemma, knowing that their default position is the punishment payoff, should prefer any other outcome to the mutually-worst outcome.

Step 1: Approve the Collateral

The first step involves the writer approving the collateral that will be associated with the SmartPiggies token and available at settlement. The writer makes a transaction with the Dai smart contract, approving the SmartPiggies smart contract to transfer Dai tokens on behalf of the writer's Ethereum account. Note that this approval takes place in the Dai smart contract.

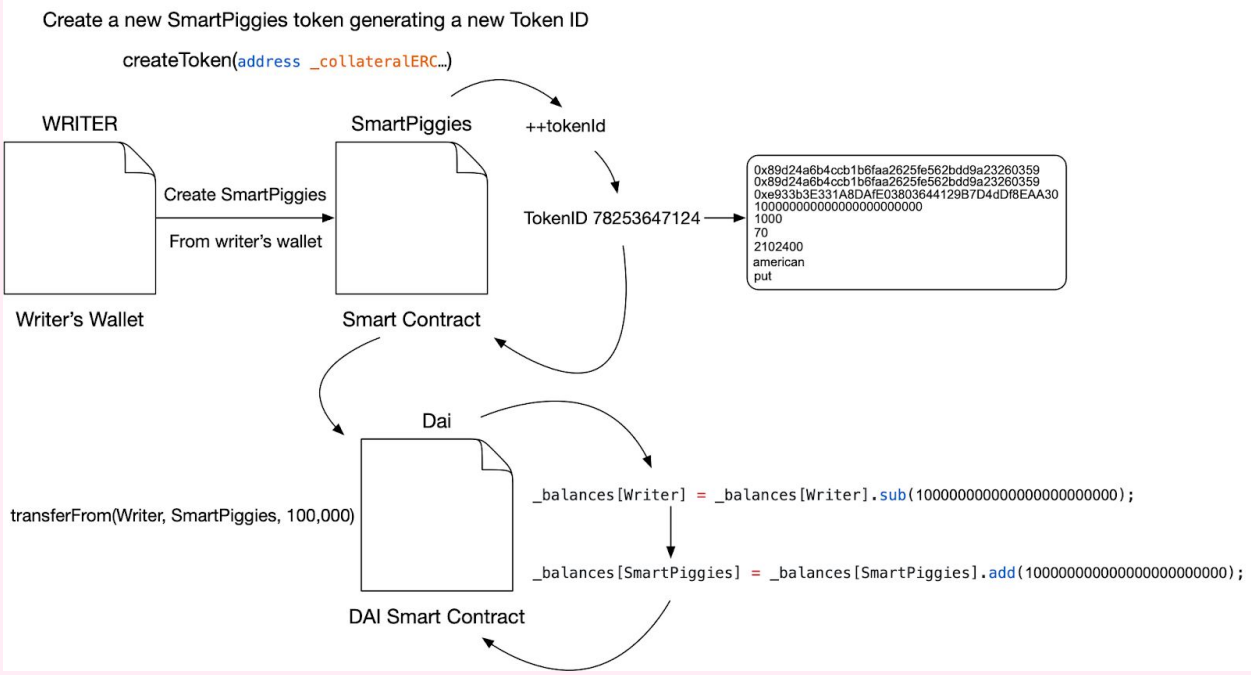


Step 2: Create a SmartPiggies Token

Next the writer makes a transaction with the SmartPiggies smart contract executing the create function supplying the required parameters:

ERC20 smart contract address for the denominated premium	0x89d24a6b4ccb1b6faa2625fe562bdd9a23260359
ERC20 smart contract address for the denominated collateral	0x89d24a6b4ccb1b6faa2625fe562bdd9a23260359
Smart contract address for the desired oracle to be used	0xe933b3E331A8DAfE03803644129B7D4dDf8EAA30
Amount of collateral to be transferred to the SmartPiggies token	100000000000000000000000
Lot size	1000
Strike Price	70
Expiration block	2102400
Option style	american
Option direction	put

In this example, the Dai smart contract's address has been specified for payment of both the premium (if used in an autonomous auction) and the option collateral. Next, the address to an oracle proxy or resolver is included. This resolver is a smart contract that will call the smart contract of an oracle service and return the data supplied by that oracle. The rest of the parameters associated with the option itself are similar to traditional financial options, with the "expiration block" serving in place of a calendar expiration date, the "option style" specified as American, and the "option direction" indicating that it is a put option.



Step 3: Sell on the Market

A SmartPiggies token representing an American put option on Lean Hog futures expiring on February 14th, 2020, collateralized up to $100,000, is now available to sell to a willing buyer. The writer of the SmartPiggies option can use the native marketing feature to sell the token to a buyer denominated in the ERC-20 currency specified in the second address parameter, in this case the same Dai stable token used for the collateral.

The writer creates a collateralized autonomous auction for the token by executing the auction function with the following parameters:

Token ID	78253647124
Auction starting price	300
Auction reserve price	100
Auction expiry	172800
Time step	5760
Price step	10

The above table denotes the SmartPiggies token ID which is designated at the time of token creation. The starting price is $300.00 and the reserve price is $100.00. The auction expiry above is written as a blocktime of 172,800 blocks, which is equivalent to 30 days.[26] The time step is set for 5,760 blocks with is equivalent to a step per day, and the price step is $10.00, i.e. everyday the asking price will decrease by $10 for 30 days.

Step 4: Purchase of the SmartPiggies Token

Once the SmartPiggies token is created and up for auction, a buyer can purchase the token for the current auction price. The prospective buyer can execute the function to satisfy the auction and take ownership of the token by specifying the token ID. The price of the token will be automatically calculated according to the auction parameters, and will be paid in Dai.

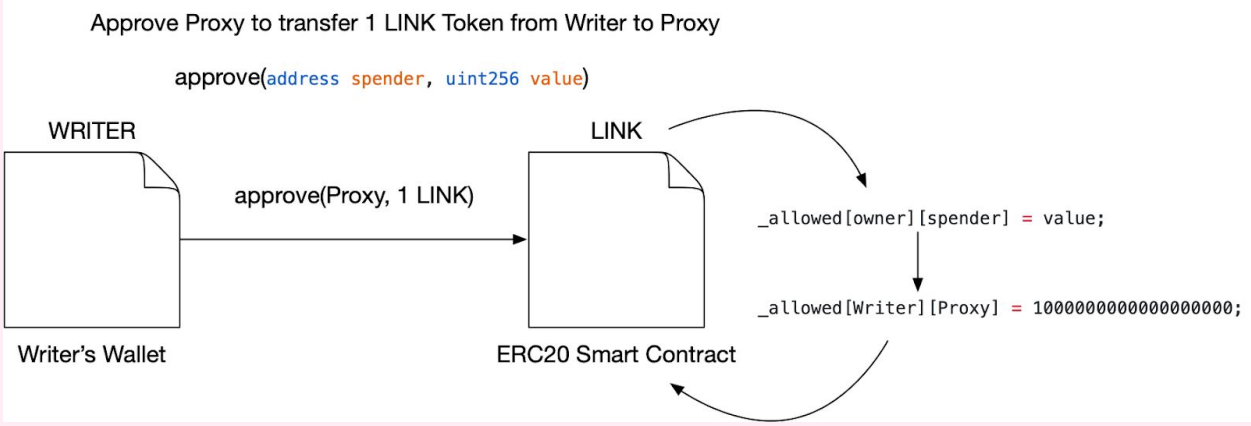
Step 5: Approve Token Transfer for Oracle Fee

The proxy address detailed in the SmartPiggies token is used to call an oracle. This proxy will use the Chainlink oracle smart contract for resolving the option settlement price. The Chainlink

[26] This is a blocktime calculation with the target blocktime on ethereum at the time of writing, which is approximately 1 block every 15 seconds, i.e. 4 blocks/minute * 60 minutes/hour * 24 hours/day * 30 days = 172,800 blocks.

service requires its oracles to be paid with LINK tokens in exchange for executing queries against external APIs. Before the oracle can be called, an allotment of LINK tokens must be approved for transfer from the calling account to the proxy. In an American style SmartPiggies option, the holder must execute this approval if the SmartPiggies option is exercised before expiration.[27]

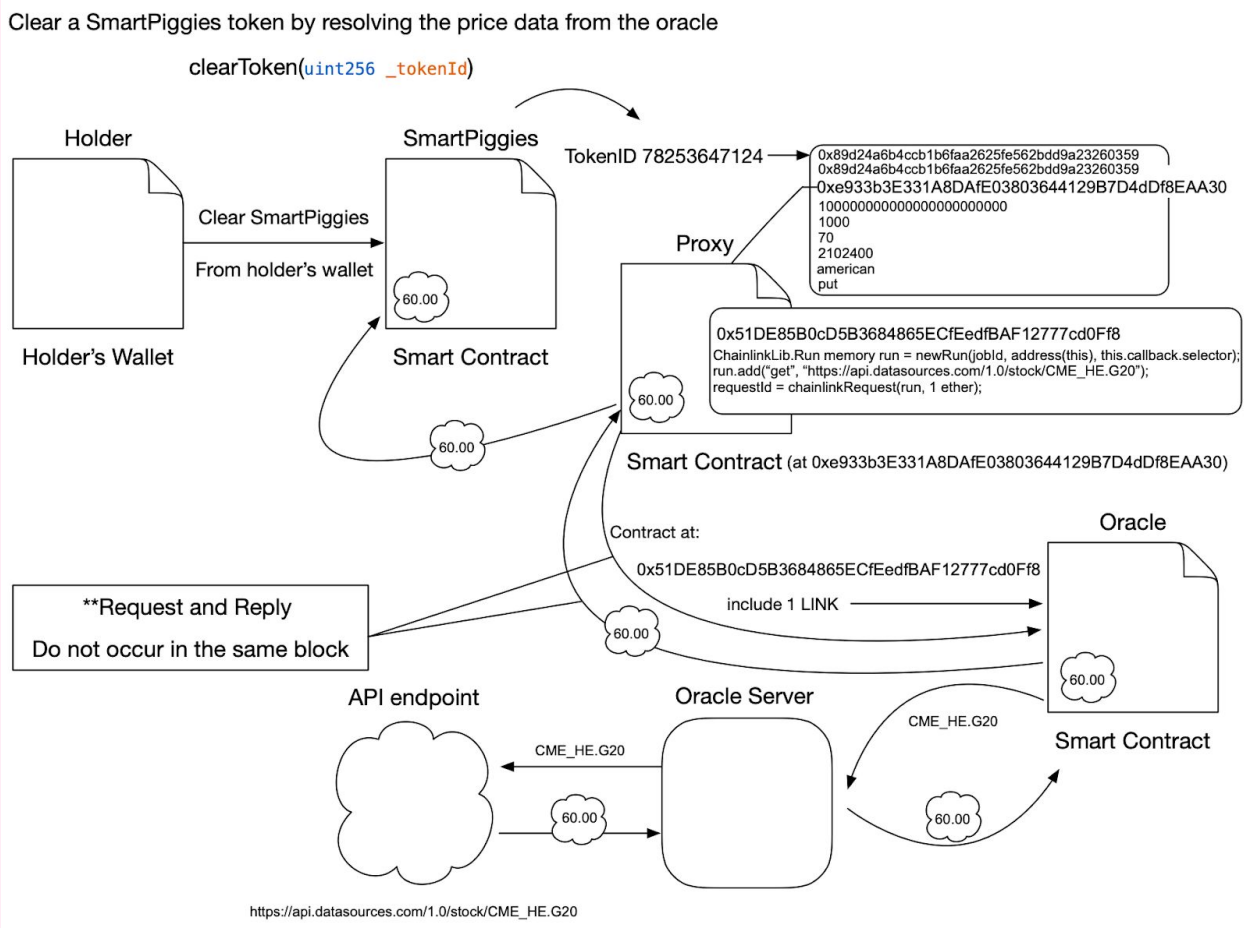
In a similar process to that outlined in step 1, the holder executes the approve function in the ERC-20 LINK token smart contract, allowing the proxy smart contract to transfer up to 1 LINK token (in this example) on the holder's behalf. This approval allows the proxy to pay for Chainlink service fees when requesting the settlement price.



Step 6: Clearing the SmartPiggies Token

After the approval for the oracle payment has been completed, the token can be cleared. In this example the holder begins the clearing process by executing a clearing function. When the clearing function is executed the SmartPiggies smart contract will call the designated proxy to resolve the price from the oracle service.

[27] If this were a European style option instead, either counterparty could settle the token after expiration; whichever account calls the oracle needs to approve the token transfer.



The proxy smart contract associated with this particular SmartPiggies token will request the price information from Chainlink. When the clearing function is executed by the holder, the proxy contract will transfer the delegated LINK token from the holder's account to the oracle smart contract as payment, and the oracle will retrieve the data at the supplied API endpoint. At an indeterminate time in the future, the oracle smart contract will return the requested price back to be saved in the SmartPiggies contract.[28]

Step 7: Settling the SmartPiggies Token

Once the price is saved in the SmartPiggies contract, the payout can be determined. Either counterparty can execute the settlement function, which will determine if the option is in the money and update the balance of collateral owed to the writer and holder within the SmartPiggies contract.

[28] There will be some time difference between the request to the oracle for price data, and the reply from the oracle with that price. Due to the nature of how blocks are added to a blockchain, a request and reply for price data may not occur in the same block, and therefore the price cannot be seen as available from a subsequent call for that data. An additional process must be executed to use the price data, after it has been returned.

The table below outlines two possible outcomes for our example that may transpire in the future, one where the option expires in the money from the perspective of the holder, and the other where it expires out of the money.

In the money	Out of the money
HE.G20 Exercise Price: <= strike price	HE.G20 Exercise Price: > strike price
HE.G20 = $60 at exercise	HE.G20 = $80 at exercise
Strike = $70	Strike = $70
Lot size = 1000	Lot size = 1000
Collateral = $100,000	Collateral = $100,000
Payout = ($70 - $60) * 1000	Payout = 0 * 1000
Payout = $10,000	Payout = $0
Balance of writer = $90,000	Balance of writer = $100,000
Balance of holder = $10,000	Balance of holder = $0

Because the SmartPiggies token was written as an American style put option, it is considered in the money if it expires with an exercise price below the strike price. In the left-hand example, the SmartPiggies token is exercised with a settlement price of $60 for Feb. 2020 Lean Hog futures, and a strike price of $70, making the token in the money with the holder gaining $10 per lot of the underlying. As the token was written with a lot size of 1,000, it will pay out $10,000 to the holder, and the remainder of the collateral will be returned to the writer. If the price of Feb. 2020 Lean Hog futures is instead $80 when the token is exercised as in the right-hand example, the token is considered out of the money, and the holder would be owed nothing. In either scenario, once the balances are updated during settlement, either party can transfer ownership rights of the collateral to their respective accounts.

Step 8: Claim Payout

The holder can claim their payout after the token is settled if the option is in the money. In the left-hand example above, the holder can transfer $10,000 worth of Dai tokens from the SmartPiggies contract to their account. The writer of the option can claim the remaining $90,000 of collateral Dai as well.

Mechanisms for Handling Off-Chain Data

As outlined in the previous section, the SmartPiggies design relies heavily on the availability of external data in order to settle created tokens. Blockchain-based systems can only authenticate data about on-chain operations; data originating outside of the blockchain network, such as prices to be used as the underlying for SmartPiggies tokens, are not subject to the same security guarantees. In Ethereum, for example, only transactions related to state changes in the network as executed by the EVM can be verified; data that did not originate from EVM operations cannot be verified in the same manner, and should therefore be considered less trustworthy.

Because accessing off-chain data is critical to whole classes of decentralized applications, providers of oracle services have started to offer a means to bridge this gap to the blockchain. However, these services come with a fundamental tradeoff, as obtaining data from outside a decentralized network in this manner raises issues of trust in the provenance of such data and concerns about possible tampering in transit to the blockchain, a system originally designed to allow for trustless transactions.[29] SmartPiggies tokens could request settlement prices from resolvers which were designed to validate signed source data, eliminating some concerns associated with oracles. Regardless of how data is brought to the chain, however, users will always be required to trust the authenticity of the data itself.

Oracles as a Bridge to the Blockchain

To provide data to decentralized applications, oracle services may either consume an external API on behalf of a requesting smart contract, or deliver data which they directly own or license, effectively acting as a broker.[30] Oracles typically have two main components: smart contracts and off-chain processes. The primary interaction with an oracle for a decentralized application is with the oracle's smart contract. This is the way a requesting smart contract needing some external data will interface with an oracle, and in this sense an oracle's smart contract is an API for accessing data through an oracle.

The second component of an oracle is an off-chain process with additional responsibilities, typically a server running a program that will "listen" to the blockchain for specific events or messages from the oracle smart contract handling on-chain data requests. When the oracle smart contract receives a request for data, the oracle's server uses the parameters passed along with the request to retrieve the desired data. The server will make requests to query

[29] See the section on "External Data Risks" for further detail on challenges and risks associated with oracle services that work to bring off-chain data on to the chain.
[30] An API is a communication protocol that allows two programs to communicate with each other.

external API endpoints, which will return data in a prespecified format.[31] When the oracle's server receives this formatted data it first parses the exact data originally requested on the blockchain, most often via extra parameters sent to the smart contract in the initial request, and then forwards the desired piece of data to the oracle smart contract. The oracle smart contract will receive this data and use a callback to send that data to the user contract that initially made the data request.[32]

Managing Oracle Trust Concerns

Third-party oracle services require a degree of trust in how those services deliver data to a blockchain. It is materially important that the service provider's systems operate as expected, free of unauthorized changes to the software or hardware that may compromise the data delivery service, and ultimately the data being provided. The entire value proposition of a data provisioning service is void if an attacker can compromise an oracle service and alter data before it is offered to the blockchain. There is thus a large reputational burden on oracle services, as users must be able to trust that those services are providing data that has not been manipulated by an outside attacker or by the service itself. A concerned party may choose to run such a service on their own as an independent participant, delivering off-chain data for a singular application. The authors view a dedicated third-party oracle service as a superior option, as an independent operator reintroduces the very trust concerns they intended to circumvent.[33]

As use of an oracle service requires reliance on computation performed outside of the Ethereum network (and thus not subject to the security guarantees that the blockchain offers), providers of oracle services may seek to offer supplemental protective measures in their stead. For example, oracle services often use a Trusted Execution Environment (TEE) to run the software requesting and returning off-chain data. This allows users to verify that the oracle server code has not been tampered with, and that the hardware environment running this code has not be compromised. While this does not provide the same security guarantees as the EVM, it may provide more assurance to users of the oracle service compared to oracle services that do not implement a TEE hardware model.

Candidate Oracle Service Providers for SmartPiggies

The authors have identified Oraclize and Chainlink as candidates for providing price data to SmartPiggies, which both use Ethereum smart contracts in concert with external servers to

---

[31] An endpoint is the receiver in a communication path, here typically a Web address that handles HTTP GET requests and returns a response with the requested information.
[32] A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action
(as per https://developer.mozilla.org/en-US/docs/Glossary/Callback_function).
[33] For example, in the case of a SmartPiggies system, an operator writing options using the SmartPiggies smart contract would have an incentive to corrupt a self-run oracle and force it to return settlement prices which were more favorable if she stood to make a large loss on a particular trade.

deliver desired data from outside the Ethereum network, though details of their implementations differ. Oraclize is a centralized service which uses “authenticity proofs” as a means of providing assurance to its users that the web API data it returns has not been tampered with.[34] Chainlink, by contrast, is a “decentralized” oracle service which allows and encourages users to query multiple oracle nodes running the Chainlink protocol to query off-chain data. Further details about the security mechanisms incorporated into each of these services can be found in the “Trusting the Chain of Custody” section below.

Oraclize and Chainlink differ significantly in their payment models, with subsequent implications for the design of smart contracts using their services. Oraclize will debit a gas fee per request for data, which is denominated in Ether. Chainlink, by contrast, requires these fees to be paid in LINK tokens. Whereas the Oraclize fee is fixed without price negotiation, the Chainlink fee is offered to the operator network. Participating Chainlink nodes will set a minimum fee for their node to provide data, or a market will develop for offering data, depending on the complexity of providing that data.[35]

One consequence of the difference in fee models for these services lies in transacting in the requisite currencies. Oraclize uses Ether, the native currency for Ethereum, while Chainlink requires LINK, an ERC-20 token. At the time of this writing, these payment methods operate differently on the Ethereum network. Oraclize requires the requesting contract to maintain a balance of Ether in the contract. A contract using Chainlink, however, can send oracle payments on behalf of the caller by way of a token transfer function,[36] allowing that contract to pay oracle fees without having to maintain a balance of Ether or LINK. A key difference in these payment methods is that one implementation requires a smart contract to hold Ether, while the other implementation allows a smart contract to avoid holding a balance of anything. This has security implications for the contract calling the oracle, which will be addressed later in the “Risks and Mitigations” section.

Third-Party Signed Source Data

As the the need for off-chain data increases, primary data sources will likely fulfill on-chain requests directly, allowing smart contracts to query data from these sources. In this model the middleware of oracles may be absorbed into these providers, and the data will have a minimized chain of custody to trust. Arguably, one need not trust a chain of custody at all, as long as the signature on the data could be verified on-chain; this benefit however comes with a

[34] Refer to the Oraclize documentation for further detail on these proofs: http://docs.oraclize.it/#security-deep-dive

[35] There are many kinds of data that can be provided, some more computationally expensive than others. Retrieving the price of an external asset is somewhat less expensive than calculating a cryptographic proof, for example.

[36] This scheme requires that an account “approve” an amount to transfer first, and then use the *transferFrom* function to transfer a token on behalf of another user/contract.

higher execution cost for verifying this signed data.[37] If the demand is high enough for existing data providers such as Reuters, Bloomberg, Nasdaq, Cboe, etc. to provide signed data to smart contracts, this may ultimately be a preferred data model.

If signed data is available from a data source, a resolver could be capable of validating the original data provider's signature on-chain, regardless of how it were transmitted to the chain. This functionality can mitigate many vulnerabilities in the data chain of custody to the extent that such signed data is available.

Markets for SmartPiggies Tokens

The section "SmartPiggies Token Lifecyle: An Illustrated Walkthrough" describes in detail how to create and use SmartPiggies tokens. Beyond this lies the question of how markets for SmartPiggies tokens may be developed in practice. This section of the paper outlines three broad market categories which could be used for exchange of tokens: peer-to-peer markets, decentralized exchanges, and centralized exchanges.[38] Use of any one of these does not preclude the others, and the authors expect that all may exist simultaneously to serve different customers who value the individual benefits that each category affords.

Peer-to-Peer Markets

SmartPiggies may be bought and sold with no additional infrastructure via peer-to-peer marketing mechanisms incorporated directly into the contract design. Peer-to-peer markets facilitated by these native mechanisms or by external auction contracts most closely resemble the OTC markets for traditional options, and are particularly well-suited to handle customized contracts offering tailored protection. While standardized SmartPiggies tokens could also be traded through peer-to-peer markets, there would be no obvious benefit in doing so versus using an exchange. The authors envision that peer-to-peer markets would be used primarily for custom SmartPiggies.

Autonomous Auctions

As described in the "SmartPiggies Functionality" section, the SmartPiggies design includes a native mechanism for sale by token holders in the form of a Dutch auction. This allows for direct peer-to-peer sale of SmartPiggies tokens without the need for any external smart contracts to manage the market for them. If token holders wish to utilize other means of peer-to-peer sale, however, they may do so using avenues such as those described below. The ability to create a

[37] As noted in section 5.2 of the TownCrier whitepaper on Hybrid TCB Minimization, "(t)here does exist a precompiled Ethereum contract to verify ECDSA signatures, but the operation requires a high gas cost." SmartPiggies could be designed to take advantage of such a signature-verification contract if signed data were available. See https://www.initc3.org/files/tc.pdf for the full paper.
[38] For background on options in traditional financial markets, the interested reader may refer to the subsection on "Development of Options Markets" in Appendix I.

SmartPiggies token as an RFP and hold a reverse auction to find a provider similarly allows for peer-to-peer purchase of a custom SmartPiggies token if a willing counterparty can be found via the auction mechanism.

External Auction Contracts

If buyers and sellers of SmartPiggies do not wish to use the native functionality for sale of the options, they may instead choose to rely on an external auction contract to manage bidding. Given the inherently non-fungible nature of SmartPiggies, these auction contracts would likely only work for sale of an existing option; without a mechanism similar to the RFP described in the “Native Purchase” section above, it would be difficult to conduct an auction for purchase (such as a reverse Dutch auction).

In general, the auction contract would be expected to escrow the SmartPiggies token for sale (entailing temporary ownership of the token), and have some mechanism for accepting bids in a particular auction format (such as an English or Dutch auction, or a blind auction using some form of commit-reveal scheme for sealed bids). The auction contract would have logic governing the sale and settlement of the winning bid in addition to execution of the particular auction type.

Sellers wishing to utilize external auction smart contracts rather than the native sale mechanism of SmartPiggies would need to be careful about the parameterization of the SmartPiggies token before locking it in escrow -- while held by the auction contract, they would be unable to destroy the token unsold if they wished to change any parameters related to the option. The seller should also have a great deal of trust in, and understanding of, the behavior of the external contract taking temporary ownership of the SmartPiggies token -- it would be possible, for example, to maliciously insert functionality into the auction contract which could invoke the token’s native sale function for a zero reserve price and effectively steal it from the seller via the auction contract. Due to additional complexity and security concerns raised by relying on external auction contracts, the authors would expect that most sellers would prefer to use the flexible Dutch auction embedded in the SmartPiggies native sale mechanism.

Decentralized Exchanges

Decentralized exchanges use smart contracts to accept bids and offers on digital assets, creating a decentralized order book and avoiding the risk of having all market participants facing a single counterparty. While the decentralized exchange model broadly provides the services expected in a traditional centralized asset exchange, a key tradeoff exists in terms of speed of order execution: given the block time in the Ethereum network, new orders would propagate only once every ~15 seconds. This block time delay, combined with public visibility of the pending transaction pool, also makes decentralized exchanges susceptible to order frontrunning, as detailed further in the “Order Frontrunning”” section below.[39]

[39] Briefly, in a frontrunning transaction, a frontrunner would observe an order to a decentralized exchange in the Ethereum pending transaction pool. Provided a sufficient spread between the buy and sell orders to

Offsetting the slower order flow and risk of frontrunning, decentralized exchanges offer users greater privacy than a centralized exchange. There is no central counterparty collecting identifying information or financial account information, and thus no store of such data which could be abused or stolen. This is a practical tradeoff with legitimate considerations for both sides, and therefore centralized and decentralized exchanges will likely exist side by side to serve the needs of different customers.

Centralized Exchanges

Centralized exchanges are operated by a single entity which takes custody of users' digital assets to facilitate higher-speed off-chain trading, and are another possible venue for trading options on Ethereum. Several such exchanges are in existence today, and in the future new exchanges specifically designed to support trading of decentralized options may be created. Centralized exchanges may also be the structure best suited for handling trading of standardized decentralized options, and a financially sophisticated entity wishing to make markets in standardized SmartPiggies tokens might develop both options contract standards and a centralized exchange platform in tandem.

Such centralized exchanges would have certain benefits compared to both exchanges for traditional options and decentralized exchanges for SmartPiggies tokens as discussed above. Compared to a decentralized exchange, a centralized exchange can offer more rapid trade execution, which potentially would result in greater liquidity and deeper order books.[40] Compared with a traditional exchange, a centralized exchange for standardized SmartPiggies tokens could offer standard products on a larger number of underlyings -- given the cash settlement nature of the tokens, there is considerably less overhead required in assembling the components for such "exotic" contracts, with the primary requirement for a new standard contract being the identification of a suitable price feed for the underlying.

A centralized exchange with a sufficiently flexible UI could also allow for markets in non-standardized SmartPiggies tokens; this would resemble the peer-to-peer markets described above, with the difference that the exchange of the option and the mechanism for price discovery (for example, bidding in an auction for a specific NFT) would be mediated through a centralized party rather than being truly peer-to-peer.

---

be matched, the frontrunner could submit two higher-fee transactions in an attempt to satisfy each side of the trade first, thereby capturing part of the spread.

[40] Centralized exchanges typically record transactions between their customers in off-chain databases which are not constrained by the block time of those blockchains. The exchanges custody their customers' assets while those assets are being traded on the platform. Customers may ultimately "settle" their trades on-chain by taking physical custody of their digital assets back from the exchange. An exchange for standardized SmartPiggies tokens could allow for trading in "virtual" tokens which were only instantiated when a user wished to move the token off the exchange.

While centralized exchanges could potentially offer users such benefits as described above, they are not without their drawbacks. Many centralized exchanges require customers to submit personally identifiable information for regulatory compliance purposes in line with their jurisdiction, which could subsequently be lost or abused.[41] Users of centralized exchanges also give up sovereign ownership and control of their assets while those assets remain on the exchange, exposing them to risk of loss or theft. These and other risks will be detailed in the following section.

Risks and Mitigations

SmartPiggies tokens, like other financial technologies, carry with them certain risks, several of which are outlined below. While some of these risks are more specifically related to SmartPiggies and cryptocurrency ecosystems, others are more general in nature and are applicable to a broad range of financial instruments.

Attacks on the SmartPiggies Smart Contract

The first logical point of concern when discussing risks is the SmartPiggies smart contract itself. If the contract succumbs to attacks directly, the security of the rest of the system supporting SmartPiggies becomes immaterial. For this reason, security of the SmartPiggies smart contract design is of paramount importance.

Honeypot Risk

A large pool of resources held by a smart contract may incentivize attackers to exploit or compromise the smart contract in order to obtain those resources. Reentrancy attacks are one well-known exploit which can be used to drain smart contracts of their Ether balances,[42] but other forms of attack are possible as well.[43] In view of these vulnerabilities, one goal of the SmartPiggies design is to avoid having the smart contract hold Ether so that it is not directly susceptible to honeypot risk. While the ERC-20 collateral token contracts referenced by SmartPiggies may still face honeypot risk, such risk is unavoidable if users wish to hedge fiat-denominated financial instruments in particular stable tokens.[44] To the extent that the

---

[41] According to market data provider CryptoCompare, most “top exchanges” require some form of know-your-customer compliance as of their October 2018 report (p.28): https://www.cryptocompare.com/media/34836036/cryptocompare_exchange_review_october_2018.pdf

[42] Reentrancy attacks are a form of recursive calling vulnerability present in the Solidity smart contract programming language. See the following link for details: https://ethereum.stackexchange.com/questions/6176/what-is-a-recursive-calling-vulnerability.

[43] For example, a vulnerability in Coinbase’s Ethereum wallet implementation did not properly reverse failed transactions when attempting to distribute Ether over a set of wallets; see the following report from HackerOne for details: https://hackerone.com/reports/300748

[44] In the event that an ERC-20 token used for SmartPiggies collateral becomes worth more than the value of the base Ethereum network (as measured by the market capitalization of that token vs. Ether), honeypot risk could increase to the extent that attackers interpreted this “imbalance” as a sign of

ERC-20 standard is well-understood and any stable token implementations built upon it have undergone thorough security auditing, those contracts are assumed to be hardened against attacks even if they present an incentive to attackers.

SmartPiggies Token Management Risks

There is the possibility that an unforeseen vulnerability in the implementation of a SmartPiggies contract could prevent users from executing the functions of the tokens properly. These vulnerabilities could include, among other possibilities, users being unable to claim the collateral from a settled option, or being unable to settle the option even if the oracle service were properly working. While the SmartPiggies design aims to reduce the need to trust counterparties as much as possible (via full collateralization and the inclusion of self-marketing mechanisms), one must at a minimum trust the smart contract code to correctly manage the tokens it issues. In the event of catastrophic failure of fundamental token management functions, the design attempts to limit risks to individual users unilaterally where possible (for example, by requiring collateralization only on a token-by-token basis rather than requiring upfront delegation of a pool of funds to draw from as tokens are created, and offering a number of fallback mechanisms to attempt the settlement of the token on-chain in the case that the external data source fails), though some risk mitigations will necessarily entail tradeoffs which must be weighed by the parties implementing the smart contract.

A critical implementation tradeoff concerns whether or not to endow the SmartPiggies smart contract with certain functionality intended for disaster recovery. This could be introduced at the level of the contract itself or at the level of an individual token issued by the contract, introducing different risks depending on which choice is made. Such functionality would typically be granted to a small set of privileged accounts in order to limit potential abuse. In one implementation, there would be no such functionality at all, eliminating the possibility of a party with “master keys” to the smart contract abusing that power to, for example, send collateral ERC-20 tokens from the SmartPiggies smart contract to personal accounts.

By eliminating this attack vector, however, the designer foregoes the possibility of allowing a virtuous overseer to stop losses on a vulnerable smart contract or otherwise limit the damage in a scenario where a catastrophic flaw was discovered. In the SmartPiggies design, the smart contract owns the ERC-20 collateral for created SmartPiggies tokens that have not yet been settled, and in a disaster scenario, it might be unable to distribute collateral to any other address with a nominal claim on it. A flawed smart contract with no disaster recovery safeguards could also continue to exist on the Ethereum network in perpetuity, with no mechanism to stop new users from invoking its flawed functions and potentially losing money as a consequence. Even if these flaws were to be publicly disclosed, and users were directed towards an alternative smart contract which had been hardened against the discovered flaw, this issue would not be fully mitigated. The authors prefer this option despite its drawbacks, as it creates an environment

---

relatively-weak security of the ERC-20 token at the base protocol layer. The authors do not consider most scenarios enabling this attack to be particularly realistic.

whereby users of the contract are assured that the deployer of the smart contract cannot directly cheat them of their collateral.

An alternate implementation could enable additional functionality such as smart contract upgradability and the capability for privileged users to control “safety valves” and/or distribution of collateral in the event that a fatal flaw with contract code were discovered. While such functionality would theoretically enable greater safety for users of the contracts under certain disaster scenarios, it would also entail a greater degree of trust in the overseers of the contract to act in the interests of SmartPiggies token holders. In addition to more direct abuse such as outright theft of collateral, an upgradeable proxy design would allow privileged accounts to point the proxy at a malicious replacement, enabling them to steal funds just as easily as they could use the upgradeability to fix a legitimate flaw. There is also the potential for existential risks during an upgrade if storage layout and design is not preserved, regardless of ill intent on the part of the parties making the upgrade; it is possible to wipe out all the data contained in a smart contract if the storage scheme is not judiciously maintained. This could result in users losing their funds, among other consequences.

A balanced approach would allow the contract to have some privileged functions for disaster recovery, but would require users of the contract to vote on activation of those functions via a decentralized autonomous organization (DAO) used to manage the contract.[45] This solution reduces, but does not eliminate, the possibility of a malicious party gaining control of the contract, and introduces a great deal of additional complexity and potential attack vectors related to management of the DAO itself and the distribution of voting rights.

Fundamentally, users must extend some trust to the implementation of any smart contract. In the view of the authors, an implementation of the SmartPiggies smart contract with no privileged functionality reduces this trust more than alternatives, though it increases the need for user awareness of the latest security developments and the need for code which has undergone extensive security auditing before being deployed and used.[46]

Third-Party Code Risks

A SmartPiggies implementation may rely upon third-party code and/or external smart contracts to support some of its functionality. Any time a programmer includes code which she did not write herself in a project, additional risk is introduced -- at the very least, the risk of misunderstanding a third-party implementation, and in the worst case the unwitting use of malicious code. Even if the programmer does write her own code, if that code is an “in-house” implementation of a reference specification, there is also risk that the implementation fails to capture the details of the specification properly, or that the specification itself contains a flaw.

---

[45] A DAO is an on-chain governance mechanism used to actively administer smart contract functionality via a decentralized voting system.

[46] A checklist of known attacks should be referred to when reviewing the code for an implementation; numerous such lists are available online, for example: https://blog.sigmaprime.io/solidity-security.html

Imported Code

Any third-party code which is imported for use potentially introduces a risk of misbehavior of the system or application which is leveraging that code. While importing code can be very convenient, and may in some cases offer additional “security” in the sense that heavily used, open-source, external libraries may have undergone thorough community review, it is still incumbent upon the developers importing that code to understand exactly how it works and what potential malign interactions exist with the system they are designing. In the case of SmartPiggies, it is possible that a third-party reference implementation for the token may be built and made available for import, or a designer implementing the specification themselves may import additional code during that process.[47] While the best mitigation for risks arising from imported code is simply for an author to write all her own code from scratch, the authors recognize that third-party code use is common and can offer such benefits as open-source review and the leveraging of prior thorough security audits.

External Contract Interaction

The SmartPiggies design as described, relies on interaction with external smart contracts for both collateral management (via ERC-20 contracts) and fetching settlement prices (via oracles). As oracles impose additional risks associated with fetching data from outside the blockchain, they will be covered in greater detail in the subsequent section on “Trusting the Chain of Custody.”

A major assumption of the SmartPiggies design is that the security and expected execution of the ERC-20 token smart contracts used for collateral and payouts will not be compromised. There exist risks in using these ERC-20 contracts as they are architected outside of the SmartPiggies design. Most or all of these contracts have been audited by third-party security firms, and have yet to fall victim to any attacks. However, a lack of successful attacks in the past is not a guarantee of future security, and any risk to these ERC-20 contracts is subsequently assumed by users of SmartPiggies tokens which reference them for collateral.

If the ERC-20 token contracts used by SmartPiggies are stable token implementations, as the authors expect may be demanded, they are subject to further risks related to the stability of the peg to their reference asset. Different stable tokens take different design approaches to this problem, and will thus have attack vectors specific to their implementation. Writers and holders of SmartPiggies tokens which reference stable token contracts for collateral must be aware of the individual risks posed to the stability of those tokens.

[47] For example, OpenZeppelin offers reference implementations of tokens adhering to standards such as ERC-20 and ERC-721, which application designers may import for use in their projects.

Stable Token Risks

Stable tokens are designed to provide 1:1 parity with a fiat currency, and enable SmartPiggies to hedge financial exposures denominated in those currencies. Some earlier ideas related to decentralized options which pay out in Ether rather than stable tokens are explored in the Velocity whitepaper,[48] and the dYdX protocol expands on these ideas to allow for cryptocurrency derivatives.[49] The availability of stable tokens greatly expands the potential for SmartPiggies tokens to hedge “real-world” prices denominated in “real-world” currencies.

The use of stable tokens, however, comes with additional risks. Stable tokens at the time of this writing typically derive their value via one of two broad mechanisms. The first mechanism is via the full backing of the issued tokens, which entails trust in an off-chain third party who bears responsibility for maintaining and securing the collateral (often a fiat currency). There exist mechanisms outside of the blockchain to verify the integrity of that third party, such as financial disclosures, audit results, possible legal ramifications, and the third party's desire to uphold their own reputation. Ultimately users of tokens backed in this manner must extend some trust to the issuer and face the risk that the tokens are not fully collateralized.

A stable token may alternatively derive its value from an algorithmic stabilization mechanism that attempts to maintain a fiat peg to the token. In this model, there exists some combination of infrastructure and incentive mechanisms to increase and decrease the supply of the stable token in response to market exchange rates such that the exchange rate tends towards the fiat peg. In this case there is a risk that the stabilization mechanism will fail, either due to faults in the modeling of incentives or failures related to the technical execution of the mechanism.

Regardless of the mechanism used to maintain parity with the reference currency, in using a stable token there is ultimately some outsourcing of trust and implicit user belief that the tokens are worth what they are claimed to be. Some stable token smart contracts also either include explicit functionality to freeze or delete tokens under certain circumstances, or are upgradeable in a manner which would allow for such functionality to be added to the smart contract in the future.[50] Should the stability of these tokens falter for any reason, or the tokens be locked or destroyed by the controllers of the smart contract which issued them, the users of SmartPiggies options could incur economic losses as the expected value of their hedges shifts or disappears altogether due to problems with the collateral.

While the authors envision stable tokens having a great deal of utility for financial applications involving fiat-denominated instruments outside of the Ethereum network, option collateral could

---

[48] See https://users.encs.concordia.ca/~clark/papers/2017_wtsc.pdf for the paper describing options referencing external price feeds.
[49] The dYdX whitepaper is available at https://whitepaper.dydx.exchange/.
[50] For example, as noted in their FAQ at https://www.paxos.com/standard/faq/, the Paxos Standard token is subject to Paxos’ “ability to freeze and seize tokens” via functionality in the token’s smart contract.

also be denominated in any other ERC-20 token in the SmartPiggies design. This may prove desirable in cases where options are being written against digital assets which can be priced directly on-chain (via decentralized exchange order books, for example), thereby avoiding risks associated with stable tokens while continuing to take advantage of other benefits offered by SmartPiggies.

Third-Party Infrastructure Risks

The most secure, privacy-conscious, and reliable way to interact with a blockchain is to run a local node connected directly to the network maintaining the chain. The greater the degree of separation between the user and the blockchain, the greater the risks are to the user. If a user cannot run a node locally, she may wish to administer a remote node, or failing that, use a dedicated node provided by a trusted third party. Perhaps least robust but most convenient is the use of third-party infrastructure and middleware to communicate with a blockchain node managed by a third party. Due to their simplicity, most users may ultimately opt to use such third-party infrastructure to communicate with a blockchain and sign transactions, but they should be aware that trusting intermediary processes and entities involves avoidable risks.

In the case of the Ethereum network, the MetaMask application is widely used to sign blockchain transactions from a web browser.[51] A user can install the MetaMask browser extension and use it as an intermediary to interact with the blockchain rather than interacting directly with her own Ethereum node. MetaMask additionally uses middleware provided by Infura in order to communicate with the Ethereum blockchain.[52] If either MetaMask or Infura were to fail or become unavailable at an inopportune time, a user may no longer be able to access a smart contract's interface, such as a frontend website connected to a decentralized application for creating SmartPiggies tokens. This may cause delays or disruptions in a user's ability to participate in the Ethereum network, which can lead to undesirable consequences during critical operations.

External Data Risks

There are many risks that come with using externally-originating data on a blockchain. While Ethereum's consensus mechanism allows users to trust data about purely on-chain transactions (such as the transfer of ownership of a particular amount of ERC-20 token or ether), this same level of trust does not apply to data originating off-chain.[53] The Ethereum network can validate

[51] See https://metamask.io/ for details on this service.
[52] Infura is a piece of infrastructure allowing remote procedure call access to the Ethereum blockchain via an API; see https://infura.io/ for details. Further technical detail on the integration of Infura into Metamask is available on GitHub at the following link: https://github.com/MetaMask/metamask-extension/blob/develop/app/scripts/controllers/network/createInfuraClient.js.
[53] This trust may be extended to transactions which refer only to on-chain data because such transactions have been agreed upon by all participants of the chain that validate data, i.e. Ethereum archival nodes which run fully validating reference software.

transactions which refer to data that originated off-chain, but cannot validate that external data itself, presenting a potential risk to users of applications like SmartPiggies which rely upon external price data for fair settlement. Provision of off-chain data to the blockchain may also comprise a chain of custody including intermediary data brokers and/or oracle services, in addition to the original source. These intermediaries add further risks which users must take into account. Finally, there is also the risk of technical failure preventing access to external data, even if no intermediary is acting maliciously and the data itself is reliable.

Validating Data at the Source

Within the SmartPiggies design, finding a trustworthy data source to serve as the reference for settlement prices is of paramount importance. This source will ultimately determine which counterparties are owed what amount of collateral at the exercise of the option. Fundamentally, this becomes a question of user trust in particular providers.[54] Due to the public nature of SmartPiggies tokens, the reference provider and endpoint will be visible to potential purchasers of a token, so they can make their own assessments of provider trustworthiness before making a purchase decision. Sellers of SmartPiggies tokens who select data providers not seen as trustworthy by potential purchasers will incur the opportunity cost of capital lockup in tokens which do not sell. The authors envision that markets (including the autonomous auctions native to SmartPiggies) will discover the providers that counterparties are willing to mutually trust.

In certain cases, additional risk mitigations are possible. If a data point to be used for settlement, such as the closing price of a particular stock on a particular day, is available from multiple sources, said sources could be queried by a specialized data resolver or oracle and results compared against one another to determine the “correct” result. For data with a single source, such as the price of a very esoteric financial instrument only traded on a single exchange globally, multiple sources could corroborate the return value of a query to that source.[55] Counterparties to a SmartPiggies token could also maintain their own infrastructure to monitor the settlement prices, though if the price they observed differed from the price delivered to the chain by an oracle, recourse would potentially be limited.[56]

If primary data sources were willing to provide cryptographically signed data, this would be a relatively strong reputation-based mitigant to bad data originating outside of the chain.[57] Additionally, such sources could operate their own smart contracts and “oracle” infrastructure to

---

[54] One might not wish to use a provider that had a short track record, was deemed to be susceptible to bribery, did not have highly redundant and available infrastructure for delivering data, etc.

[55] A sufficiently powerful and flexible oracle service could theoretically offer aggregation of values across multiple external sources outside of the SmartPiggies core design. Such a service could also work as a “meta-oracle,” aggregating queries of yet other oracle services against a single external API endpoint to make sure that they agreed.

[56] An aggrieved counterparty could of course make a point to never use that oracle service and/or data source again in the future, but might be unable to be “made whole” for the transaction that prompted discovery of failure to deliver proper data.

[57] A business with a reputation to uphold as a trustworthy data provider should not want to provide signatures for bad data.

answer queries for their data on-chain with no intermediaries. Though the authors are unaware of any large data providers which currently offer such services, it remains an interesting theoretical possibility, and the SmartPiggies design is intended to be flexible enough to take advantage of such services if they were to be offered.

Trusting the Chain of Custody

Oracles facilitate the provision of off-chain data to blockchain operations, but they also carry a number of risks and increase the need for users to rely on trusted intermediaries. As previously noted, the data that is provided by an oracle cannot be verified by a blockchain validation process because the blockchain does not “know” anything about this data, let alone how it should be validated. Although oracles cannot verify the source data itself, they may be able to offer some assurances that the data was not tampered with in transit from the source to the blockchain, either in collusion with a counterparty to a SmartPiggies token or out of malice. Though the data source cannot be fully trusted, risks associated with the way data was provided to the blockchain can be minimized.

The “Candidate Oracle Service Providers for SmartPiggies” section of this paper describes two different oracle solutions which allow for verifying the data chain of custody via a combination of secure connections and trusted hardware. Additional details of the security measures taken by each service to mitigate trust concerns are described below.

Oraclize Security Model

Oraclize takes advantage of the TLSNotary service to verify the state of the website that produced the data and defend against attacks occurring between the Oraclize servers and the websites being queried.[58] Oraclize also uses an attestation scheme called SafetyNet developed by Google, which provides for a verifiable, trusted execution environment.[59] A proof is derived from this scheme to validate the software being run by the service and the hardware on which it is running. This generated proof is then compared to a previously calculated proof stored at an IPFS pointer (this is stored off-chain for execution cost considerations); if the proofs match, the services are assumed to be valid, and the data accurate or at least trusted to have been delivered as it was found by the Oraclize service at the endpoint.

Chainlink Security Model

To facilitate its decentralized oracle service, Chainlink provides oracle node software that anyone can operate as a service provider on the Chainlink network. An operator stakes LINK tokens (Chainlink’s native ERC-20 token) as a collateralized guarantee that she will perform the jobs requested of her node, and will furthermore have an individual reputation associated with her node. A requesting smart contract can designate which nodes to use for data requests,

[58] TLSNotary is a technology designed to cryptographically prove the presence of a particular HTTPS web page in a browser; see https://tlsnotary.org/ for details.
[59] See the following documentation for further details on the SafetyNet attestation scheme: http://docs.oraclize.it/#security-deepdive-authenticity-proofstypes-android-proof

allowing them to choose more reputable operators and/or those staking more LINK for more critical data requests. If an operator accepts a request but does not fulfill it, a penalty deposit of her LINK tokens will remain locked indefinitely, and if she provides faulty data, or inconsistent service, her reputation will decline.[60] Because Chainlink is an oracle service in which there are many operators to choose from, requesting smart contracts may ask for data from many oracle nodes, thereby decreasing the dependency on any single response and potentially increasing the quality of the returned data.[61] However, all Chainlink nodes will be running the same code, and if this code itself is vulnerable to some attack, nodes may all report the same inaccurate data when queried. Chainlink nodes can be run in a trusted execution environment (TEE), further reducing the required amount of trust in the operations of the oracle service. Chainlink incorporates the Town Crier "authenticated data feed system," which leverages Intel's Software Guard Extensions (SGX) to provide off-chain data to smart contracts.[62]

Trusted Execution Environment Risks

While trusted hardware would appear to mitigate certain risks in the data chain of custody, these TEEs themselves become potential attack vectors. Oraclize takes advantage of Google's remote attestation technology called SafetyNet for its TEE, and Chainlink leverages Intel's SGX. While Google and Intel are large companies with substantial reputations to uphold, reputation alone does not guarantee the integrity of critical infrastructure systems which could be compromised by software vulnerabilities, physical vulnerabilities, or government authorities. There are also questions about the protections TEEs offer as attacks and compromises are continually being identified. At the time of writing runtime attacks against SGX still appear to be possible.[63]

Regardless of the guarantees provided by various TEE or other security schemes, off-chain data itself cannot be verified in the same manner as data which originated entirely on-chain. Introducing any external data to a blockchain-based system, regardless of security measures taken in its delivery, introduces a measure of trust back into a system originally designed to be trustless. It is important to note here that while both counterparties to a SmartPiggies token must trust the oracle or service used to settle it via external data, they need not trust each other.

Data Availability Risks

When fetching data from outside a blockchain, there is the possibility that intermediary infrastructure (such as an oracle's backend servers) may be down, or that the API endpoint itself may be dead. Such issues may occur even if the oracle smart contract is functioning

---

[60] See the Chainlink documentation for a description of these penalty deposits: https://github.com/smartcontractkit/chainlink/wiki/Protocol-Information#penalty-deposits
[61] NYC Ethereum #67 Meetup: Securely Connecting Smart Contracts to Off-chain Data. Dec. 3, 2018 NYC Future Labs.
[62] See the Town Crier whitepaper for further details on this system: https://www.initc3.org/files/tc.pdf
[63] See, for example, the code-reuse attacks described in the following paper: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-biondo.pdf.

properly and the external API is properly specified. Because the SmartPiggies design relies on external data for price settlement, and signed source data is not readily available at the time of this writing, these availability risks should be taken into consideration by SmartPiggies token users.[64]

In the event that an oracle's backend servers are unresponsive, the callback responsible for delivering the requested data may never return. Even if the clearing process were reinitiated, data may only return when the oracle's servers come back online. Specifying an oracle reputed for high availability may be one way to mitigate against this risk, and if a single endpoint serving the desired data is down, making multiple oracle calls to different data sources may mitigate this issue. If the API endpoint is permanently unresolvable, the SmartPiggies token cannot be settled unless a token settlement resolution mechanism is in place. The section on "SmartPiggies Functionality" describes a number of resolution mechanisms which could be directly incorporated into SmartPiggies to facilitate settlement of a token for which the endpoint had become unavailable. These include a mechanism for the counterparties to agree to change the endpoint, a mediated third-party resolution, and a form of on-chain negotiation via bidding on a split of the collateral.

If none of these resolution mechanisms were used, a SmartPiggies token could be transferred from the holder to the writer, who could then use the collateral reclamation function to claim the full collateral amount. In this situation the counterparties would need to communicate ahead of time to agree on the transfer and manual resolution of the settlement. Depending on the performance of the option, it would be incumbent upon the writer to manually settle the collateral distribution if a payout was owed to the holder. Due to the lack of strong incentives for a writer to transfer the payout to the holder in this scenario, the authors believe that most counterparties to a SmartPiggies token will use one of the various settlement resolution mechanisms built in to the token instead.

Another potential mitigant for risks of an unavailable API would be to query the API periodically throughout the life of the option, with an agreement to use a value or aggregation of values captured before exercise in the event that the API were no longer live at the time of exercise. The Velocity whitepaper explored this idea with a custom solution; a sufficiently powerful oracle service could also potentially act in a similar role.[65]

API Risks

As the API endpoint is a critical field ultimately determining the value of a SmartPiggies token to the counterparties, users of SmartPiggies tokens need to be aware of risks surrounding the API

[64] Signed source data could also become unavailable, but would be able to be brought to the chain via any channel as long as the signature were valid, mitigating some of the intermediary risk present with oracle infrastructure.

[65] The "PriceGeth" service in the Velocity design would publish external currency pair data to Ethereum on a near-real-time basis. Publishing a price every block (as described in the design of PriceGeth) may not be necessary to achieve resolution of a SmartPiggies option with an unavailable API endpoint.

itself. These risks may be related to the URL specifying the endpoint, a compromise of the service residing at the endpoint, or a user interface for entering or reading the endpoint field from the smart contract.

Deliberate Specification of a Malicious API Endpoint in the Smart Contract

An attacker may attempt to cheat the purchaser of a SmartPiggies token by specifying an API endpoint for settlement which would resolve to a pre-arranged price from a malicious or colluding data source. The attacker could attempt to find a buyer for a SmartPiggies token which referred to a malicious URL that had been pre-specified by the attacker, or she could attempt to use a copy/paste attack to entice a seller into creating a token with a faulty URL that appeared "safe" to the victim. Due to the importance of the API endpoint in resolving the settlement price for a SmartPiggies token, users need to be careful if pasting these endpoints, and should attempt to verify the integrity of a URL before specifying it during token creation or before purchasing a token.

Tools could be built for the specific purpose of allowing users to verify the correctness and reputation of candidate API endpoints to use in conjunction with SmartPiggies tokens. Such tools might exist in the form of a user interface integrating a token-curated registry (TCR) of API endpoints or resolver addresses.[66] The incentive mechanics of a TCR are beyond the scope of this paper, but broadly speaking a TCR could act as a "trusted" on-blockchain source for dispensing API endpoints, avoiding the need for their manual specification.

Compromise of API Endpoints via Structural Attacks

Even with mitigants such as a TCR for reputable settlement endpoints, the risk of a compromised API endpoint is still present, and to some extent is an irreducible source of risk due to the nature of the internet itself. Despite preventative measures such as SSL certificates and encrypted TLS connections, certificate authorities can still be compromised,[67] and DNS servers routing traffic to the data providers themselves are also susceptible to attack.[68] Any oracle services taking advantage of these technologies to increase the security of their offerings are ultimately susceptible to the same attacks, though it may still be preferable to use an oracle which makes secure connections to external data over one that does not. Users of SmartPiggies tokens need to be aware of these risks, and should take as many preventative measures as reasonably possible when employing these technologies, even if those security precautions are not completely foolproof.

---

[66] A token-curated registry is a smart contract that keeps a cryptographically secure record of information, with curators of that content effectively staking reputation on its quality and/or veracity. See the following link for an explanatory overview of TCRs: https://medium.com/@simondlr/city-walls-bo-taoshi-exploring-the-power-of-token-curated-registries-588f208c17d5

[67] For example, the certificate authority Comodo was compromised as discussed at the following link: https://bravatek.com/comodo-certificate-hack-it-gets-worse/

[68] The web-based wallet provider MyEtherWallet was subject to this form of DNS attack as described in the following article: https://www.coindesk.com/150k-stolen-myetherwallet-users-dns-server-hijacking

Misrepresentation of Smart Contract Data by a Malicious UI

It should also be known that the UI of a decentralized application for creating or transacting in SmartPiggies tokens could be faulty, error prone, or outright malicious in representing the smart contract data. Although a smart contract is unable to misrepresent its own data, a front-end application designed to interact with it could lie about that data. This is a subtle but important point: the very tool which allows easy access to and broad participation in a decentralized application may misrepresent the interactions it makes with an incorruptible data source.

Users should ensure they are employing a highly reputable UI to the extent possible. The strongest current mitigation for UI attacks is to observe the data representing a SmartPiggies token directly on the blockchain by running one's own full node. In practice, one may also wish to use multiple UIs such as individual block explorers to cross-reference on-chain data.

Attacks on Decentralized Networks / Blockchain Risks

Due to the nature of SmartPiggies as a decentralized application running on a blockchain-based network, it is important to consider attacks against the network itself, and risks that users may face due to the inherent nature of how blockchains function.[69] These risks range from miner attacks to privacy risks, and risks involving sovereign ownership of cryptographically-secured assets.

51% Attacks

Proof-of-work blockchains such as the current implementation of Ethereum are subject to the possibility of an entity or collective which controls a sufficient proportion of the available hash power (used to create proof-of-work consensus) to rewrite part of the chain's history. An entity able to successfully carry out a 51% attack would be able to double-spend tokens and temporarily censor transactions by reorganizing the main chain. In the SmartPiggies design, this could be problematic for any mechanism to market and/or transfer a specific SmartPiggies token.

For highly secured networks, 51% attacks are expensive to sustain and can only persist on a temporary basis. The attackers would have to target specific transactions that occur during the attacking period, and once executed, a significant block reorganization would raise alarm globally. Due to the targeted nature of the attack and the immediate public knowledge of its occurrence, 51% attacks are less likely to impact users than generally believed. The mitigation for 51% attacks is to wait out the attack while it is being sustained, and to avoid accepting any transaction as final until it has been confirmed at a sufficient depth in the chain such that it would be effectively impossible for an attacker to rewrite. In the SmartPiggies design, any

[69] Because Ethereum is the first such network that the authors are aware of which supports all of the functionality required to implement SmartPiggies, it will be used in several examples below; however, many of these risks would also apply to other blockchain-based systems.

auction result, or transfer of a SmartPiggies token or collateral payment, should not be treated as finalized until the transaction has been sufficiently confirmed in this manner. By using the autonomous auction capability of SmartPiggies tokens or some other escrowed mechanism, users can also protect themselves from trivial double-spend attacks, though they will need to wait for auction results to be sufficiently confirmed.[70]

Hard Forks

Blockchain structures are subject to the risk of "hard forks" where two or more versions of the chain with incompatible consensus rules diverge from one another, effectively becoming "new" blockchains after the fork for as long as they are individually maintained (by miners, for example). The primary issue that arises in a hard fork is determining which units of value recorded on the separate versions of the blockchain are the "true" units of value. This is something of a philosophical question well beyond the scope of this paper, but for a design such as SmartPiggies, some more tractable considerations can be made.

Given the nature of SmartPiggies as a form of collateralized option, the ancillary question of "which fork does the collateral have value on?" becomes relevant. This will be determined by some combination of factors including market prices and mining power accorded to each of the forks, as well as which chain the ultimate issuers of assets used as SmartPiggies collateral choose to honor.[71] As long as all forks continue to support the settlement functionality of SmartPiggies options, SmartPiggies tokens created before the fork may be settled on all forked chains, and new tokens may be created on any of the forks. In the event that one or more forks no longer support the full set of functionality that the SmartPiggies design requires (such as the ERC-20 tokens used as collateral, or oracle services used to fetch settlement prices), the value of SmartPiggies tokens on those forks may be expected to tend towards zero.

Network Availability and Congestion

The use of SmartPiggies assumes that users can access the underlying decentralized network. However, in some use cases for SmartPiggies, access alone may not be enough and speed of transaction processing may be paramount (e.g. when calling a transaction to settle an American-style option). While the Ethereum network has relatively fast block times compared to some blockchains (such as the Bitcoin blockchain, for example), there may be delays in transaction processing for a number of reasons. If a large number of nodes on the network are

---

[70] In the trivial case, an attacker would agree to send a SmartPiggies token to another user in exchange for a payment in Ether, without using escrow. The attacker would broadcast two transactions to transfer the token, one sending it to the buyer as agreed, and the other sending it to another address controlled by the attacker. If the buyer sent payment after seeing the transaction made to her address, the attacker could then attempt to get the second (conflicting) transaction accepted into the longest version of the chain. If successful (via a 51% attack, for example), the attacker would then retain both the SmartPiggies token and the Ether payment made for it.

[71] For example, if collateral were in the form of an ERC-20 token backed by physical fiat currency held in a bank vault, the issuer of that token might only allow redemption for tokens on a single chain after a hard fork.

experiencing a distributed denial-of-service attack, there may be delays in propagating transactions. If the network itself is being flooded with transactions, there may be lengthy queues before a particular transaction gets processed. Fees will also likely rise during times of network congestion, and while a user may attach a very large fee to their transaction in the hopes of faster processing, lower processing time is probabilistic rather than guaranteed even for large-fee transactions.

Users of SmartPiggies for time-sensitive purposes such as American-style options should be aware both of the inherent processing time limitations of the underlying network, and the possibility of delayed transactions and/or high fees during times of high congestion. The ability to voluntarily pay a high fee is a partial mitigant for congestion scenarios, but unless blockchain-based network architecture changes significantly to allow greater transaction throughput in the future, this is simply a limitation that SmartPiggies users must be willing to face.

Privacy Risks

Many decentralized blockchain-based networks are public by design, meaning that parameters of all SmartPiggies tokens such as counterparty addresses, underlying, collateral amounts, etc., will be publicly visible to any interested party. As such, SmartPiggies tokens may be better suited to hedging than speculation, as any information that an arbitrageur is looking to exploit will be publicly hinted at in their trade.

The public nature of the network may be beneficial to users working to build a reputation as a broker of SmartPiggies, making it easier for them to find counterparties if they operate from a consistent set of addresses. If users wish for their trading activities to remain private, however, they may need to seek options on a different venue, as this is generally a platform-wide limitation of public blockchain networks.

Sovereign Ownership / Private Key Risks

Sovereign ownership of cryptographically-secured bearer assets such as SmartPiggies tokens allows for the elimination of intermediaries, but introduces a new class of security issues for the owners of those assets. As on-chain assets are controlled via cryptographic signatures, it is absolutely necessary for owners to ensure that they do not lose control of the private keys which produce these signatures. If the holder of a SmartPiggies token lost the private key controlling the account that held the token, she would no longer be able to claim any collateral that she were due from the settlement of the token, for example.

Mitigations for risks related to private key management include not leaving keys on an internet-connected device where they may be stolen by an attacker, using multiple addresses with different private keys to hold smaller amounts of cryptographically-secured assets, or using multi-signature wallets requiring m-of-n keys to sign for transactions. While there is no absolutely foolproof mechanism for avoiding loss or theft of private keys, a combination of

prudent user behavior and technical understanding of the means available to secure them is of paramount importance.

Order Frontrunning

The proof-of-work security model currently used in Ethereum relies upon miners to process transactions, which are gossipped across the network and included in potential blocks at the discretion of the miners. From an economic perspective, one may assume that miners will attempt to preferentially include higher-fee transactions first, and to the extent that this is the case, it exposes transactions to the possibility of a frontrunning attack wherein the attacker "jumps in front" of another transaction she observes on the network by submitting a mutually-exclusive transaction with a higher fee attached.

Frontrunning would primarily pose a risk to users trading SmartPiggies tokens on external decentralized exchanges. Once a bid or offer is submitted to the decentralized exchange, that transaction becomes publicly visible in the Ethereum pending transaction pool. Any party able to observe a spread between limit orders in the decentralized order book and new orders arriving in the pending transaction pool may attempt to capture that spread (net of fees) by submitting offsetting orders to the decentralized exchange with higher fees than the order in the pool, fulfilling the order in the book and then immediately transacting with the party which had just submitted their lower-fee order. This is a probabilistic attack but nevertheless a real concern in using decentralized exchanges.

The autonomous auction facilities of SmartPiggies are subject to frontrunning to a lesser degree. A frontrunner might be able to satisfy an active auction for a particular SmartPiggies token ahead of another user who had already submitted a transaction to do so, but if she wanted to try to capture a spread with her attack, she would then have to resell the token, incurring fees and potentially losing the interest of the original customer at an increased price. For this reason, the authors expect that any user of SmartPiggies tokens worried about the risk of frontrunning on decentralized exchanges would prefer to use the autonomous auction facility of SmartPiggies instead.

Block Stuffing Attacks

In a block stuffing attack, an attacker submits a number of high-fee transactions sufficient to fill an entire block,[72] forcing anyone who wants a transaction to be processed to either wait or offer very high fees in an attempt to jump in front of the attacker's transactions.[73] With respect to SmartPiggies, a practical example of such an attack would be a case whereby one counterparty

[72] The maximum gas limit for a single block of transactions in Ethereum is currently in the vicinity of 8 million gas.

[73] This attack is not just theoretical; an attacker was able to successfully carry out a block stuffing attack against the "Fomo3D" gambling application, making off with over 10,000 ether as a result. Onur Solmaz presents a detailed explanation of this attack on his website for the interested reader: https://osolmaz.com/2018/10/18/anatomy-block-stuffing/

had sold an American option which had become well in-the-money some time after the sale. If this counterparty stood to lose a significant amount of collateral were the holder to exercise the option, she could attempt to maintain a block stuffing attack long enough for the price of the underlying to move against the holder of the option. This could be very expensive, both due to the probabilistic nature of the attack succeeding, and also the inherent uncertainty regarding where the price of the underlying could move while sustaining the attack. Theoretically, either counterparty should be willing to spend up to the amount of collateral that they could potentially reclaim to attack or counterattack; this form of attrition leaves one counterparty with nothing and the other with a loss depending on who wins the transaction race, which may discourage the attack from being undertaken in the first place.

Et Alia Attacks

In Ethereum, the EVM itself always poses a risk in being compromised by some unforeseen behavior; low-level operations of other blockchains may pose similar risks. These may come from a misunderstanding of how the EVM operates or a zero-day exploit.[74] A misunderstanding in the operation of the EVM is somewhat more palatable to consider than an EVM zero-day, as there are much larger consequences for the latter.

One example of how a misunderstanding of the EVM can lead to smart contract security vulnerabilities is the case of the Fomo3D application on Ethereum.[75] The creators of Fomo3D attempted to reduce the risk of cheating by disallowing smart contract addresses from transacting with the lottery contract, with the idea that if only “real people” deposited money from their wallet accounts, no one would be able to craft extra attack behavior that could cheat the game. To implement this defensive measure, the architects of the Fomo3D lottery contract incorrectly assumed that the code size of a smart contract was never zero.[76] However, this assumption failed to take into account the behavior of smart contract constructors, which run only once when a smart contract is deployed and return a code size of zero. This oversight allowed an attacker to place the entire execution behavior of a smart contract into its constructor, bypassing the restriction on addresses interacting with the lottery contract and ultimately enabling them to claim the entire contract jackpot.[77]

The case of Fomo3D is a single example of latent attack vectors waiting to be exploited in the EVM; many others are surely possible and will arise in new contexts in the future. The

---

[74] A zero-day exploit is a vulnerability in computer software that is unknown to those interested in mitigating such a vulnerability, i.e. the software vendor. These vulnerabilities can be used by attackers to exploit the software until such time as the vulnerability is discovered or disclosed to a larger community. The day the vulnerability is learned of is the zero-day: https://en.wikipedia.org/wiki/Zero-day_(computing).
[75] The smart contract governing the application was a classic lottery scheme designed to reward a jackpot winner after collecting funds from the many participants hoping to win.
[76] This was seemingly sensible, as deployed contracts will indeed have a nonzero code size, unlike externally owned accounts.
[77] For further detail on how this misunderstanding was exploited, refer to the following article: https://medium.com/quantstamp/what-we-learned-from-fomo3d-part-1-2c316db3d1e1

SmartPiggies smart contract or any external contract that the design relies upon could be subject to and even specifically targeted by such attacks. The authors expect that thorough audits as well as wide peer review of popular open source projects will generally mitigate this risk. Should there be a widely catastrophic problem for the EVM itself, the community as a whole may come together to resolve the issue. It should also be noted that the SmartPiggies smart contract intentionally avoids functionality that could be used to compromise it as a whole, and each individual SmartPiggies token is designed to be a separate instance with restricted permissions mostly accessible only to its owner and writer.

Economic Loss

Economic loss is a risk with any financial instrument, and SmartPiggies tokens are no different in this regard. Losses may be incurred due to unforeseen market conditions, flawed pricing models, or misunderstanding of the instruments being traded. As a hybrid of several novel technologies, SmartPiggies tokens are subject to all of these risks, with some subtle differences compared to their options counterparts in traditional financial markets.

Loss Due to Market Conditions

An investor in any financial instrument is always subject to risk of loss due to adverse market conditions. For SmartPiggies tokens, extreme volatility in the underlying could make the exercise of an American option difficult to time, or could simply allow the price of the underlying to shift further away from the strike in either direction. New information entering the marketplace which was unavailable at the time the option was sold can similarly expose holders of SmartPiggies tokens to risk of loss, either on the option settlement or on the premium if they attempt to sell the token to another investor before expiry. Risk of this type of loss is uneliminable in practice; it can be potentially reduced with high-quality information and pricing models, but some portion of this risk is generally considered to be irreducible in financial markets.

Loss Due to Pricing Models

Options pricing models are quite complex, typically involving multiple stochastic components and requiring sufficient data about historical price movements in the underlying. SmartPiggies pricing models would need to make similar calculations, presumably with additional adjustments for lockup of capital as collateral, risk of transaction delay, inherent blockchain time delay in fetching external data, blockchain and/or oracle fees, and so forth. A model which did not properly capture the relevant parameters for pricing a SmartPiggies token could result in loss due to overpaying or undercharging for the premium relative to the theoretical risk of the option. Thoroughly researched and backtested pricing models would help mitigate this risk; the specification of such a model is beyond the scope of this paper, however.

Loss Due to User Misunderstanding

Traditional options are not a simple financial instrument to begin with, and SmartPiggies differ from them in subtle ways. Users should always have a firm understanding of the instruments they are using, whether SmartPiggies or otherwise.

Misunderstanding of Options

Any user of a SmartPiggies token on either side of the trade should be aware of the type of instrument they are using. While SmartPiggies tokens differ in certain ways from traditional options, they are similar enough in their basic structure and execution (particularly with regard to the calculation of their payout at exercise) that users should be sure to understand how options work before creating or purchasing SmartPiggies tokens.

A SmartPiggies token is a derivative instrument, and therefore can only be valued in conjunction with the price of the underlying on which it has been written. If a user believes a SmartPiggies token to be similar to an ERC-20 token with a (theoretically) intrinsic value, or if they mistake the amount of collateral associated with a SmartPiggies token for the value of the token itself, they are misunderstanding how the option works. This misunderstanding could lead to mispricing on the part of a writer, and uneconomical bids on the part of a buyer, which could in both cases lead to economic loss which might otherwise have been avoided.

Misunderstanding of SmartPiggies

If a user of SmartPiggies does not understand the nature of the interconnected parts of the system, she exposes herself to risk of loss as a result. SmartPiggies tokens may often be highly illiquid due to their custom nature, and should not be confused with potentially more liquid tokens such as ERC-20s which are fungible by design. There is also a strong need for users to examine and trust the data source which will determine the price of the underlying at settlement as well as the infrastructure (such as an oracle system) which will be used to deliver that price to the SmartPiggies smart contract. As noted in various sections above, there is also an inherent lag in blockchain-based systems due to the block time such that settlement of American-style options cannot be processed instantaneously. If a user of SmartPiggies is familiar with traditional financial options with potentially unlimited payoff, they may be surprised to receive only the collateral associated with the token rather than a larger amount they believe that they are due. To mitigate risk of loss arising from such misunderstandings, users of SmartPiggies should learn thoroughly about the technology and how it differs from their understanding of traditional financial options before they create or purchase SmartPiggies tokens.

Regulatory Risks

As with many digital assets, there is an open question as to whether regulators in any particular jurisdiction would choose to identify SmartPiggies tokens as regulated financial products. The

uncertainty associated with regulators' opinions on this matter could result in negative material impacts to a business model or market.

While SmartPiggies inherits the global, decentralized, permissionless nature of the Ethereum network, users who reveal to the public their dealings in SmartPiggies tokens may find themselves subject to regulatory action in the jurisdictions in which they reside. Some jurisdictions may impose significant requirements on users of digital assets, such as know-your-customer, anti-money laundering, regulatory reporting, and licensing requirements, or in some cases outright bans on transacting in those assets. Even if users do not advertently divulge their usage of technologies such as SmartPiggies, decentralized networks such as Ethereum are pseudonymous rather than anonymous, and activity deemed illegal by regulators may be traced to particular users regardless.[78]

Punishments for violation of regulations vary in severity by jurisdiction, and users of digital assets must make themselves aware of the laws and regulations to which they are subject. Regulators have already pursued certain individuals for providing a platform that facilitated direct trading between customers in the case of EtherDelta, which was found to be operating as an unregulated securities exchange despite never having taken custody of any assets.[79] Other regulators have expressed the opinion that the ideation of any code or process that might foreseeably result in existing regulations being bypassed should render the ideator liable for all activities enabled by that idea, which would have obvious implications for creators of decentralized financial applications.[80]

Aside from SmartPiggies tokens themselves being banned, it may be the case that that tokens or cryptocurrencies critical to the operation of SmartPiggies may be subject to sanctions. For example, Ethereum itself may be banned or the collateral tokens themselves may become a target of regulatory scrutiny. Even if a token itself is not targeted by regulators, the ecosystem that supports it may be, as was the case with the Basis stable token.[81] To help mitigate the risk of regulatory action, users of SmartPiggies tokens should make themselves aware of any applicable regulations in their particular jurisdiction, though they should also be aware that the interpretation of legislation and subsequent regulatory actions may shift unpredictably.

---

[78] For example, tools like Crystal Blockchain (https://crystalblockchain.com/) are designed to help investigators easily track transactions and potentially correlate real-world identities with blockchain activity.
[79] For details of the charges, see https://www.sec.gov/news/press-release/2018-258. This case is particularly interesting as it begs the question of what constitutes a securities exchange. By design, EtherDelta was established as a decentralized peer-to-peer exchange and the creator arguably only provided a medium of communication between individuals.
[80] See, for example, https://www.cftc.gov/PressRoom/SpeechesTestimony/opaquintenz16.
[81] See the following article for further details on the dissolution of Basis due to regulatory concerns: https://techcrunch.com/2018/12/13/basis-backed-with-133-million-from-top-vcs-to-build-a-price-stable-cryptocurrency-says-its-shutting-down-and-returning-the-money/

A Final Note on Risks

While this section has attempted to describe a broad spectrum of feasible attacks and associated risks to users of SmartPiggies tokens, it cannot be considered an exhaustive list. Different decentralized networks and different implementations of the SmartPiggies design on those networks will all be exposed to idiosyncratic risks of those platforms, and new attack vectors may surface at any time. As with any other technology or financial innovation, users must take care in their dealings and avoid exposing themselves to the possibility of unacceptable personal losses.

Regulatory Considerations for SmartPiggies

While regulations potentially pose risks for users of SmartPiggies tokens, there are legitimate considerations for why regulators would want to oversee a complex financial product such as a decentralized option. Traditional financial markets contain a number of risks which regulators attempt to minimize, some of which are applicable to SmartPiggies and some less so. At the same time, there are opportunities for regulators and legislators to enact mutually beneficial policies which would serve the aims of the countries they represent and the users of certain digital assets alike.

The Necessity of Existing Options Regulation

Traditional options have historically been highly regulated. This is due to a number of intrinsic market-based risks:

- High leverage: Due to the leveraging effect of options, writers can expose themselves to far greater liabilities than they are able to cover with cash they have on hand.
- Unlimited liability: Since there is no upper bound on the price of an asset, the writer of a naked call option is theoretically exposed to unlimited losses.
- Opaque funding: Due to the contract nature of traditional options and inherent complexity of accounting, it is difficult if not impossible to know whether a counterparty will have funding available at the time a contract is exercised.
- Correlation risks: Even with the best of intentions, a counterparty that writes options may incur several large losses at the same time, creating liabilities that far exceed existing funds.
- Limited liquidity / price manipulation: Due to the specificity of individual options contracts (varying degrees of freedom in underlyings, counterparties, strikes, maturities, etc.) it is difficult for liquid markets to develop. Without active market-driven bids and offers, investors can be subject to price manipulation by other market participants, their counterparties, or exchanges themselves.

- Systemic financial risks: Counterparties may depend on hedging instruments which themselves may be subject to the above risks. An extreme event can cause a systematic cascade of failures across many counterparties.

In addition to the above market-based risks faced by users of traditional options, it has been necessary for regulators to protect customers from conflicts of interest and potential fraud. Option writers are incentivized to collect premia up front, and absent any legal requirement to the contrary, they have no incentive to ensure that they will be able to cover their potential liabilities. In practice, investors have been quoted unfair prices, seen prices manipulated, had positions prematurely closed out, or had funds stolen by unregulated service providers.[82] As traditional options are effectively legal agreements, it should not be a surprise that they require a reliable regulatory environment and legal system to function as intended.

How Decentralized Options Mitigate Regulatory Concerns

Whereas traditional options are legal agreements and require regulatory and/or legal enforcement, SmartPiggies tokens inherit the autonomous, trustless nature of the networks on which they exist and are therefore much less reliant on external enforcement to function as intended.

Mitigating Intrinsic Market-Based Risks Through Collateral Guarantees

In the SmartPiggies design, the smart contract has full control of the funds that are associated with issued tokens, with gains and losses limited to the collateral controlled by a particular token. As such, there is a greatly reduced risk that the counterparty to a SmartPiggies token could fail to deliver funds.[83] With SmartPiggies tokens there are no issues with overcommitted writers, unlimited liabilities, correlation risks, or systemic financial failures.

Mitigating Conflicts of Interest Through Transparency and Network Guarantees

The SmartPiggies design offers a number of transparency and network guarantee benefits to users, which are also potentially advantageous to regulators. The collateral associated with SmartPiggies tokens can be proven to exist, and is directly observable in the balances of the associated ERC-20 smart contracts. Similarly, the option terms and execution logic for any particular SmartPiggies token are directly observable and immutable by design, as is the data source which will be used to settle the token.[84] Both counterparties to a SmartPiggies option, as

---

[82] See, for example: https://www.investor.gov/investing-basics/avoiding-fraud/types-fraud/binary-options-fraud.

[83] Under the assumption that the software has been properly implemented and that the external data determining the settlement price is available, there should be zero counterparty risk.

[84] The only mutable field associated with a funded SmartPiggies token is the address of the external settlement price data resolver, which may only be changed with the agreement of both counterparties. All other parameters of the token are fixed for the duration of its existence.

well as relevant regulators, have the assurance of the network's guarantees of proper execution, as well as a complete and permanent audit trail. Additionally, as SmartPiggies tokens exist on an open network with complete transparency, to the extent that the tokens' autonomous auction mechanism is used, regulators would have a complete and real-time view of global market activities, events, and sentiment.

Aligning National Strategy and Open Decentralized Networks

The network of technologies built on private infrastructure which comprises the internet allows for public access to communication around the world, and with the advent of public cryptocurrency systems such as Bitcoin and Ethereum, anyone with an internet connection can now transfer value over these networks. Whereas regulators are charged with protecting the public users of these technologies, legislative bodies define rules and mandate how regulators should enforce good behavior on these networks.

Regulators may struggle if these technologies develop faster than legislative bodies can mandate, as novel technology may enable behavior that has never existed before. In some cases, lack of revised directives and guidance has created pressure on regulators to apply pre-existing mandates in emerging areas of technology with potentially undesirable consequences. With respect to cryptographic assets built on open decentralized networks, regulators may be inclined to approach their duties in the same manner as they would currently-regulated financial assets. However, alternative approaches to regulating new technologies on these developing public infrastructures may offer unique national strategic opportunities. These technologies have the potential to enable greater business opportunities, economic growth, new demand for national currencies, and national resilience, but these opportunities may be compromised if regulators are not given the appropriate legislative mandates.

Economic Opportunities Presented by Decentralized Options

A strategic priority for any nation state is to attract capital investment. By providing a legal environment conducive to the development and use of digital assets traded on open decentralized networks, nation states with attractive business environments can reduce frictions and attract capital from a wider audience. Open decentralized networks have the capacity to make advanced financial services more broadly available to small businesses which have traditionally been unable to access them, and to remove costs, middlemen, and frictions from the market as a whole. Decentralized options such as SmartPiggies tokens, for example, could be used by a small farmer or manufacturer anywhere in the world to hedge against price fluctuations in their inputs and products in cases where those values have been too small or specialized for traditional finance to support. Via open decentralized networks, small businesses that are typically engines of growth and employment may gain access to financial products that help them manage their risks and subsequently drive economic activity.

While large businesses are already well-served by the current financial system, they may also reap some benefits from SmartPiggies as a uniquely efficient and trusted way to establish an OTC agreement. Rather than establish a traditional banking relationship with its associated costs, a large business could pseudonymously place their SmartPiggies token offer on the global market for bidding.

Due to its autonomous auction mechanism, the SmartPiggies design allows for markets to operate at a global scale which is likely to result in a more competitive and transparent execution compared with traditional broker or banker models which are fundamentally limited in reach and scope. Any market that excludes those who would otherwise participate is likely less efficient as a result, as buyers and sellers alike would have access to fewer potential counterparties. With SmartPiggies, by contrast, everyone has access to a single market.

Reconciling Competing Incentives Between Regulators and Users

Regulators and users of decentralized options such as SmartPiggies may have different desiderata for features of the technology, particularly with respect to ERC-20 stable tokens used as collateral. Users would likely prefer a stable token implementation with few or zero intrinsic controls (meaning that the smart contract issuing those tokens is unable to censor their use in any way once issued), and would likely be drawn to stable tokens in strong national currencies, particularly global reserve currencies. Regulators may be likely to share a preference for their own nation’s currency to be considered globally desirable, but may also prefer to have stronger controls on digital stable tokens pegged to their currency in order to enforce compliance with anti-money laundering laws as well as other regulations. Both regulators and users of stable tokens pegged to national currencies should be aware of these inherently divergent incentives, and the broad range of potential compromises between them. Regulators and legislators in particular jurisdictions will ultimately have the final say on what is legal in their jurisdictions; users similarly will have the ultimate choice about where to direct their financial activity on the partial basis of which jurisdictions support stable tokens that they wish to use.

Regulation of Data Providers

The largest external dependency in the SmartPiggies design is the set of data sources the system relies on for settlement. Users may prefer to use regulated data providers that would face legal consequences in cases of customer abuse, manipulation, or hacking. A government body that enforced customer-friendly regulation on officially sanctioned data providers would enhance the market appeal of products and companies covered by such data providers, such as decentralized derivatives.

Strengthening National Economies via Resilient Infrastructure

As national economies have become increasingly dependent on digital technology and infrastructure, they have become increasingly vulnerable to electronic attack, both by criminals and by state-level entities. Attacks vary from the compromise of sensitive information to denial

of access to critical services, or even direct theft or destruction of value. Centralized information repositories or control systems may offer a concentrated incentive to attackers, and may also be intrinsically more susceptible to certain forms of attack. Nations that are capable of functioning with infrastructures and economies that largely avoid centralized points of control and information would be less vulnerable to such attacks. Open decentralized networks and systems such as SmartPiggies built atop that infrastructure would benefit from the resilient design of the internet itself, and provide services that would offer greater availability and user protection in the face of attacks on those systems.

In the short-to-medium term it is unlikely that open decentralized networks will garner widespread economic adoption, with few real economic use cases being deployed atop them as of this writing. Decentralized financial applications such as the price insurance offered by SmartPiggies, which can help generate real economic value for users, may serve to incentivize broader adoption of these networks, and this adoption could help speed the deployment of subsequent decentralized tools which drive economic value creation.

While SmartPiggies tokens would inherit the resilience of the open decentralized networks they operate on, they would still be individually dependent on the stable tokens they reference as collateral. While there exist various approaches for issuing stable tokens, the most popular as of this writing appears to be issuing a token against actual fiat currency held in an audited bank account, with the promise of fixed-rate redemption of tokens for fiat on demand. Some research into government-issued stable tokens has also been pursued, though this could potentially result in an extreme point of centralization and vulnerability depending on the design of the system. In order to take greatest advantage of the resilience of an open decentralized network, any system built on top of it should retain the topology of decentralization as much as possible.

Governments wishing to pursue a maximally-decentralized digital economy may wish to encourage private companies to issue stable tokens in their national currency with regulatory oversight rather than issuing those tokens themselves, much as central banks in countries such as the United States have chosen to leave money creation to private banks. This would offer several structural advantages, such as diversification of risk of stable token failure and delegation of responsibility for operational security across many private issuers rather than a single government issuer. Governments wishing to retain control over the tokens themselves could still continue to do so via regulation of private issuers, allowing the same visibility and access that they would have had they issued the tokens themselves.

Future Directions

The core SmartPiggies design offers developers and users a large degree of flexibility, and as such, opens up a number of possible future paths for development and expansion of the system. These paths lead in many directions: deeper ecosystem integrations with Ethereum or other decentralized networks, alternative implementation strategies, development of markets for

SmartPiggies tokens, standards development for endpoints, and more. A number of possibilities are outlined in this section, and the authors anticipate that many useful additions may be pursued beyond those described below.

Ecosystem Integrations

The SmartPiggies smart contract by design relies on external smart contracts to support its functionality. Future developments in on-chain data provision services, stable tokens, and cryptocurrency wallets could enable SmartPiggies tokens to serve a broader market with potentially improved performance and reduced cost.

Advances in On-Chain Data Provision Services

Given the general usefulness of data originating off-chain to various classes of decentralized applications, the authors anticipate a number of developments around the provisioning of data to the blockchain. Improvements in this area would be advantageous to implementations of SmartPiggies to the extent that they streamlined data delivery architecture and increased the quality of external price data available on-chain. New oracle services may offer enhanced security assurances, persistent data availability solutions, and aggregation over multiple independent data sources. Well-known data providers or first-party sources could provide data directly (or indirectly via signed data) to on-chain applications, potentially bypassing the need for third-party oracle services entirely.

Alternative Stable Token Implementations

Stable tokens targeting a peg to the same underlying currency will see competing implementations, which will offer users of SmartPiggies more choices for collateral. A number of alternatives already exist on Ethereum for USD-pegged tokens, and additional implementations may be developed in the future.[85] Different implementations take different approaches to maintaining their peg, such as off-chain collateralization with dollars, on-chain collateralization with cryptocurrencies, or seigniorage models. Alternative proposed implementations have described the use of an algorithmic mechanism rather than collateral, though these remain untested in practice.[86] As various stable token designs proliferate and new designs are tested, users of SmartPiggies tokens will benefit from a greater breadth of choice in the manner in which this important component of the overall system is handled.

---

[85] Maker Dai, Gemini Dollars, Paxos Standard Token, and USD Coin are various ERC-20 implementations pegged to the US dollar.

[86] Basis (https://www.basis.io/) was a project attempting to create an “algorithmic central bank,” and Ampleforth (https://www.ampleforth.org/) is an alternative elastic supply algorithm which appears designed to avoid some of the regulatory issues which caused the Basis project to shut down.

Stable Tokens Supporting Additional Regions

As designs for stable tokens are further explored, it is possible that a number of new tokens pegged to other international fiat currencies will be issued by various entities. Stable token implementations for popularly traded currencies such as the Euro and Yen are already being pursued,[87] and if successful, their methods may be applied to bring other tokenized currencies to Ethereum or other blockchains.[88] In the longer term, it is possible that certain governments or central banks will issue "digital fiat" tokens, which are a topic under active research.[89] New stable tokens pegged to various regional currencies around the world would allow users of SmartPiggies to make and request hedges in those currencies to offset local exposures, greatly broadening the addressable market for options as a risk mitigation tool in those areas.

Mobile Wallet Integrations

As non-fungible tokens such as SmartPiggies become more common and their utility more widely understood, mobile wallets may begin to provide support for their standards. This would increase the potential base of users and the convenience with which SmartPiggies tokens could be transacted. Mobile wallet software with particularly well-functioning UX/UI would be greatly beneficial in this regard: if the complexity of a SmartPiggies token could be made tractable on a mobile phone screen, it would benefit mobile users and market makers in general.

Ethereum Standards Integration

Emerging standards for the Ethereum network may offer additional benefits to users and developers of SmartPiggies implementations. The Ethereum Name Service, for example, could simplify the process of finding, sharing, and verifying particular data source proxy implementations, as well as making it easier to interact with the SmartPiggies smart contract itself. EVM packages which allow smart contract developers to reuse and upgrade code on-chain could increase the security and efficiency of the system. In addition, a SmartPiggies Ethereum Improvement Proposal to codify the interface, would offer developers a clear way to maintain consistency and interoperability with wallets or other smart contracts working with decentralized options.

Market Development

If SmartPiggies tokens prove to be a useful financial product, new tools for trading or venues in which to trade them may arise, or existing markets may be expanded to facilitate trading of

---

[87] See https://stasis.net/ for an example of a Euro-pegged stable token on Ethereum, and https://www.gmo.jp/en/news/article/789/ for an announcement regarding a Yen-pegged stable token.
[88] The facility of "wrapped collateral" across blockchains could allow for Ethereum implementation of stable tokens issued on other blockchains, similar to the implementation of wrapped BTC: https://www.wbtc.network/
[89] See, for example, China's work towards standards for digital fiat currencies: http://africa.chinaDaily.com.cn/a/201812/08/WS5c0b159da310eff30328fd61.html

SmartPiggies tokens. Given the autonomous marketing facilities of the tokens, the initial versions of these tools could be explorers at various levels of sophistication, which would scan a blockchain network for available tokens and RFPs currently being auctioned. Various decentralized exchanges could also facilitate trading of SmartPiggies tokens with greater flexibility for setting the transaction prices, and centralized exchanges might also wish to list SmartPiggies tokens. In this latter case, the authors expect that such exchanges may also act as market makers, potentially listing standardized options in the manner of Cboe in the US.[90] Different exchanges in different regions could offer markets for options on local securities, allowing traders to speculate on financial instruments that they were familiar with, or to more easily buy protection on local investments.

Standards Development

It is possible that the widespread availability of decentralized options may lead to the demand for standardized products which could be traded more fungibly on exchanges, and/or demand for standardized sets of endpoints which could be relied upon for certain underlyings.

Standardized Contracts

As noted above, standardized options contracts may be an appealing product for some customers. These may be used primarily for leveraged speculation, or for simple hedging on common exposures. Standards could be developed by an exchange or a third-party interface which only allowed for certain deliberately-limited sets of parameters to be used in the creation of SmartPiggies tokens. Such standardized tokens would then become fungible with one another, potentially facilitating greater liquidity. While the authors envision the flexibility of SmartPiggies encouraging more users to write and request more specifically-tailored options, standardized options may be an easier “introductory” product for some users to understand, and may develop into their own speculative markets.

Endpoint Registries

Endpoint registries are a slightly different form of standardization: rather than specifying a set of possible parameterizations for all aspects of a SmartPiggies token, such a registry would instead provide a standardized mapping of underlyings to known endpoints which could fetch the settlement prices for those underlyings.[91] This could be a centralized service offered by a third party such as an exchange, oracle, or data broker, or a decentralized service offered through a mechanism such as a token-curated registry, with reputation and deposited capital at stake for the curators. Such a registry would likely be an implicit or explicit component of standardized contracts as discussed above, particularly if those standardized contracts were issued by an exchange.

---

[90] Cboe itself could feasibly be one such exchange, of course.

[91] In the current SmartPiggies design, this would be a mapping of underlyings to oracle resolvers designed to fetch prices for those underlyings.

Even for users who did not want to limit themselves to specific lot sizes, strikes, expiries, etc., a standardized list of “tickers” for different underlyings could come to be a convenient offering, as it would effectively offer a Schelling point for buyers and sellers who wanted to write an option on a specific underlying. This could be particularly convenient in cases where the counterparties did not know each other and were using autonomous markets to transact in the option -- rather than needing to discover through initial trial and error which endpoints for particular underlyings were satisfying to market participants, a standard endpoint could be used by both. While this potentially introduces concerns about trusted third parties, depending on who curates the registry, such a registry could also offer a great deal of convenience to users of SmartPiggies.

Alternative Implementation Strategies

The design of the interface specification for SmartPiggies is fairly open-ended and extensible. While the authors envision a single smart contract which can create American and European call and put options, these may be split apart into more efficient contracts, which are either managed in concert or can stand on their own. Specific implementations which attempt to facilitate “true” barrier options matching the collateral associated with SmartPiggies tokens may also be constructed. More complex contracts could utilize core factory contract(s) to create “one-click strategies” such as straddles or ladders.[92] Different implementations may also hardcode interoperability with specific oracle services, or allow for flexible selection of an oracle at the time of token creation. Feasibly, different styles of auctions for the autonomous marketing functionality could also be implemented. As long as these different implementations follow the specifications for core functionality of creation, transfer, collateralization, etc., the options tokens created from those contracts would be able to be used in any venue that supported the SmartPiggies token standard.

Partially Collateralized Options

One possible extension of the concept of SmartPiggies involves the creation of option tokens which are collateralized by a pool rather than a specific amount of capped collateral per token. This would allow for tokens to behave more similarly to traditional (non-barrier) financial options in that there would be no theoretical cap on the potential payoff for any particular option issued against that collateral aside from the amount of collateral in the pool itself. While this would break with the core SmartPiggies design, such partially collateralized options would be less expensive (as the capital cost for maintaining the collateral could be made lower for the same amount of notional claims), and might therefore be attractive for some investors.

[92] Such contracts might have to be managed by a third party in order to ensure fulfillment of all required components of the strategy; in practice, an exchange acting as a market maker could facilitate this kind of contract.

Such a partially collateralized system introduces obvious risks and attack vectors in a pseudonymous environment such as the Ethereum network. Chief among these is that a counterparty writing options against a pool of collateral could easily write a single option to herself under a different pseudonym with terms that would quickly issue the holder an unlimited profit. After collecting premia from writing many other options against her pooled collateral, she could exercise the one she had written to herself, draining the entire pool and absconding with the premia from holders who now had no recourse to claim collateral with their own options.

Certain limitations could be put into place in an attempt to ameliorate the above issue: systems could be constructed which limited the range of inputs for partially collateralized options such that it would be “impossible” to write an option that would instantly grant an unlimited payoff to the holder; collateral could be subject to lockup periods proportional to the amount to be withdrawn, allowing holders with legitimate claims to withdraw before a fraudulent transaction could drain the entire pool, or such lockups could be made “elastic” as more and more collateral was withdrawn against a particular claim; multiplier caps on some standardized amount of collateral per options could be set, with a known premium adjustment made for each multiplier to allow customers to pay a greater premium for a greater potential payout. Each of these potential fixes is likely subject to other workarounds, or introduces new issues of its own (for example, the case of a legitimate claim on a large portion of the pooled collateral by a speculator who made a bet on a “black swan” event that did in fact materialize, or pricing distortions caused by difficulty in modeling the cost of capital lockup). One other possibility is that entities which would offer such partially collateralized options would only be specially licensed and/or regulated entities, with a national government or other regulatory body overseeing their activities, and a reputation at stake in the event that attempts to cheat or defraud investors were discovered. While such a solution runs counter to the ethos of decentralized, “trustless” blockchain systems, it could be a possibility nevertheless.

Alternatively, a pool may be government-controlled or sponsored. By contributing to a pool of collateral (potentially with seniority status), a government could choose to subsidize the risk of investing in industries which it considered a national priority. Rather than directly funding industries with cash grants and tax incentives, governments could leverage global investor capital to efficiently invest in national priorities by limiting the risk faced by those investors. By not directly picking winners and bypassing intermediaries and decision makers, the purity of the government’s intent could be made clear, as government subsidies have historically been susceptible to corruption and fraud. By serving as a partial guarantor, a government could achieve its ends without spending any funds in a fully transparent way.

Conclusion

The primary use case for SmartPiggies tokens in the view of the authors, stated simply, is “price insurance for everyone.” The flexible nature of the design and the facility of cryptographic stable

tokens to represent regional currencies appears poised to enable financial risk management for a wide variety of users who have not had access to the financial protection provided by options. Users in regions where there are no options exchanges could be empowered through decentralized options such as SmartPiggies to offset risks to their local businesses, or to hedge investments in local financial markets. Cryptocurrency users wishing to hedge their various token holdings could use SmartPiggies to do so, which may be an appealing use case given the high price volatility of many cryptocurrencies and limited derivatives available to hedge them in existing markets.

A concept which the authors have found helpful thinking about use cases for SmartPiggies is the “long tail of market access” -- while certain existing financial products and services do serve particular markets fairly well in terms of providing risk management in the form of options or other derivatives, many people globally are not served by such. Rather than seeking to provide yet another financial product targeted towards those users who already have a number of similar tools at their disposal, SmartPiggies aims to fill in gaps in the long tail of alternative use cases outside of those currently provided for in markets such as the United States. The authors also anticipate that users who have been marginalized under the present global financial regime will find additional use cases beyond what traditional options markets currently offer, specific to their individual situations and financial needs.

References

Antonopolous, Andreas. *The Internet of Money*, Merkle Bloom LLC, 2016.

Biondo, A., Conti, M., Davi, L., Frassetto, T., & Sadhegi, A. “The Guard’s Dilemma: Efficient Code-Reuse Attacks Against Intex SGX,” 2018.

Damodaran, Aswath. *Investment Valuation: Tools and Techniques for Determining the Value of Any Asset*, Third ed., John Wiley & Sons, 2012.

Eskandari, S., Clark, J., Sundarsesan, V., & Adham, M. “On the Feasibility of Decentralized Derivatives Markets,” 2017.

Hilpisch, Yves. *Python for Finance*, O’Reilly, 2015.

Juliano, Antonio. “dYdX: A Standard for Decentralized Margin Trading and Derivatives,” 2017.

Nakamoto, Satoshi. “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2009.

Zhang, F., Cecchetti, E., Croman, K., Juels, A., & Shi, E. “Town Crier: An Authenticated Data Feed for Smart Contracts,” 2016.

Appendix I: Options in Traditional Financial Markets

This appendix is intended to give the reader a background on the mechanics of options in traditional financial markets, as an understanding of such is prerequisite to understanding the SmartPiggies design. If the reader is already familiar with financial derivatives, this content may be skipped without loss of context.

Options Explained

In the simplest terms, an options contract is very much what it sounds like: an option to transact in a particular financial asset at some point in the future. Because options rely on a reference asset (called the "underlying"), they are said to be a type of "derivative" asset – they derive their value from the value of the underlying. Options are distinct from other derivative assets such as futures due to the optionality inherent to the contract: the holder of an option has the right, but not the obligation, to execute the transaction for the underlying asset at some point in the future (this is termed "exercising the option").

Options Components

An options contract has a few important components which must be understood to grasp the utility of the contract as a whole: underlying, lot size, strike price, expiration date, "style of exercise" (i.e. American vs. European), and whether the option is a put or a call. All of these components are specified at the creation of the option, and do not change during the life of the contract. These shall be described individually below, followed by a more holistic example to clarify how they act in concert.

Underlying

As briefly described above, the underlying is a financial asset (separate from the option itself) to which the option contract will refer. This underlying will be the asset which may (optionally) be transacted between the counterparties if and when the option is exercised.[93] As a concrete example, the financial asset may be shares of a particular company such as Apple. The option contract would then refer to the price of Apple stock (designated on the NASDAQ exchange under the ticker symbol of "AAPL"), from which it would ultimately derive its value. Further discussion on pricing will follow; for now it is sufficient for the reader to understand the connection between the option and its underlying, and the fact that the option must have an underlying asset to have any meaning at all.

[93] This is in the case of physical settlement, where the “physical” underlying, such as shares of stock, changes hands. Cash settlement, more akin to the settlement of SmartPiggies tokens, only has the counterparties exchanging cash differences that would be due if the physical underlying were exchanged and then immediately sold in the marketplace.

Lot Size

The lot size is most easily thought of as a "multiplier" on the price of the underlying. Concretely, the lot size specifies how many units of the underlying would be transacted at exercise of the option. The option may be created (or "written," as the act of creating an option for sale is often termed) on any number of shares of AAPL. For exchange-traded options, there are standard lot sizes, but for OTC options any lot size may be specified depending on the needs of the purchasing counterparty. For example, if AAPL were currently $100 per share an option could be written to transact a single share, or 1000 shares – in the latter case, the underlying would be $100,000 worth of AAPL stock at the initiation of the contract.

Strike Price

The strike price is a benchmark price for the underlying which the actual price will be compared to at the time of exercise to determine the value of the option. When the option is exercised, the current price of the underlying will very likely deviate from the strike price. Depending on which direction it deviates in, and the "directionality" of the option itself, the holder of the option may stand to benefit from the exercise. Exactly how they accrue this benefit will be described in the holistic example following the definition of these components.

Expiration Date

The expiration date is the date at which the option may be exercised if the holder chooses to do so. The expiration date is always in the future relative to the time at which the option is written. It is the date on which the strike price and the price of the underlying would be compared to determine the value of the option to the holder at the time of exercise. There is an exception to this, which is related to the "style of exercise" discussed below.

Style of Exercise

The exercise of an option may be in one of two "styles:" European and American. A European option may only be exercised at the expiration date. An American option, however, may also be exercised at any point before the expiration date. A European option therefore may be thought of as a special case of an American option, though the additional optionality offered by the American option changes its relative value even on the same underlying.[94]

Direction of Transaction

Lastly, an option may have one of two "directions," which relate to the "direction" (buy or sell) that the holder of the option will have to transact the underlying with respect to the writer of the option. An option contract where the option holder may purchase shares (at the strike price) from the writer is termed a "call" option – the holder of the option may "call in" the shares from the writer of the option at the specified strike price if they wish to do so. An option where the

[94] These would obviously collapse to the same value if the American option were held until the expiration date.

holder would be able to sell shares at the strike price to the writer at the time of exercise is known as a "put" option – the holder may "put" the shares back on to the proverbial shoulders of the writer.

A Practical Example of an Options Contract

It may be increasingly clear to the reader how these components work in concert at the time of exercise, but for the sake of completeness, an example putting all the pieces together:

- Underlying: AAPL (assume the current price is $100, for simplicity)
- Lot size: 10 shares
- Strike price: $150
- Expiration date: 1 month into the future (on a trading day)
- Style: European
- Direction: Call

The owner of this option will have the right, but not the obligation, to purchase (call option) 10 shares (lot size) of AAPL (underlying) from the writer one month from the date the option is written (expiration date), but not before (European option), for a price of $150 per share (strike price). On the exercise date, the holder of the option will observe the price of AAPL in the marketplace. If that price is below the strike price, their call option will be worthless – exercising it would only allow them to purchase shares from the writer at an above-market price, which no rational actor would ever do. In this case, the option is said to be "out of the money." However, if the market price of AAPL is above the strike price, the option is "in the money" and the holder accrues a benefit: if she exercises the option, she can acquire 10 shares of AAPL at a below-market price. She could subsequently sell those shares into the market for a profit (with the lot size acting as a multiplier on her profit), or she could elect to hold the shares and accrue the benefit of a lower price paid.

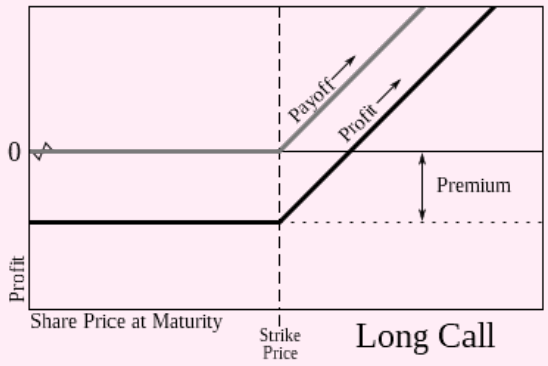
If the reader understands the example above and how the benefit accrues to the holder at the time of exercise, they will be able to envision other cases of options (American style, put options) and how the payoffs may accrue to the two counterparties in those cases.

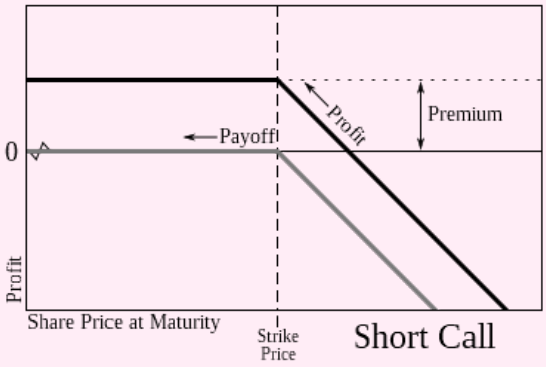
A Description of Standard Options Contract Payoffs

The diagrams that follow illustrate the payoffs for the four “vanilla” options positions, where a single option is written (as opposed to a multi-option strategy which will have more complex potential payoffs). The owner of a call option is considered to have “long call” exposure, and if the price of the underlying at maturity exceeds the strike price, she would see a (per-lot-unit) payoff equal to the amount that the maturity price exceeds the strike price.[95] The option seller,

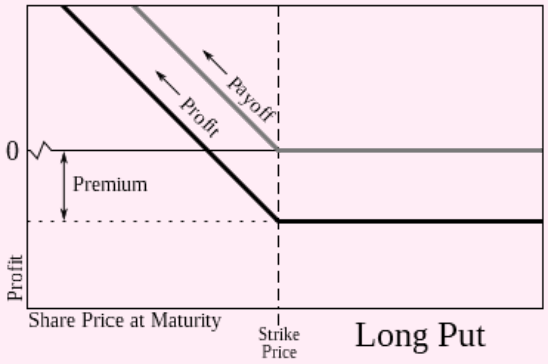
[95] This is the case described in the AAPL example above.

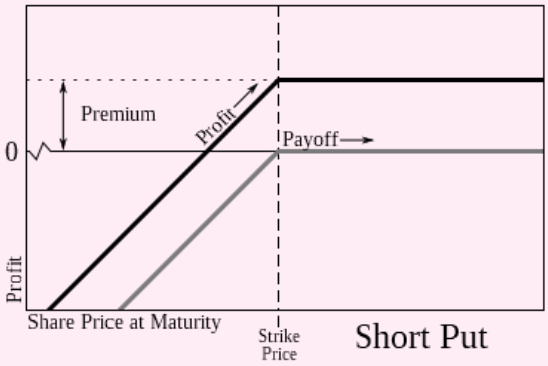
by contrast, has “short call” exposure and would incur a loss equal to the amount that the maturity price exceeds the strike price.





The owner of a put option has “long put” exposure, and if the price at maturity drops below the strike price, she would see a positive payoff equal to the amount that the maturity price falls below the strike price. The option seller with “short put” exposure would incur a loss equal to the amount that the maturity price falls below the strike price.[96]





By combining calls and puts at various strikes and expiries on the same underlying, an investor may construct more complex strategies with different payoff diagrams. These four vanilla payoffs, however, remain the foundation of those strategies, and are analogous to the four cases included in the SmartPiggies design.

A Note on Options Pricing

A natural question of pricing may arise once the structure of an option is understood. This price (or "premium" as it is most often termed) is what the writer charges the buyer for the right to potentially make a transaction in an uncertain future. Precisely due to that uncertainty, options pricing is a difficult modeling problem, and a full description is well outside of the scope of this

[96] All option payoff diagram images available under Creative Commons license from https://en.wikipedia.org/wiki/Option_(finance)

paper.[97] SmartPiggies tokens will require valuation models as well, which may be based on traditional options pricing models; the authors envision that external markets will demand and create such alternative models to account for the specific nature of SmartPiggies tokens.[98]

Options Use Cases and Market Development

The ability to use options as a means of hedging a position in other financial assets provides a novel risk management tool to investors. Under classical financial assumptions, investors in assets such as stocks or commodities take exposure to the risk of price changes with the belief that markets are efficient and any relevant information on the factors which will contribute to the growth of that asset are reflected in its market price. The real world very often fails to behave in this manner, however, for any number of reasons. The market may have failed to adequately price in the risk of certain events due to unevenly distributed information, or risk derived from "unknown unknowns" (now commonly referred to as "black swan" events) may materialize, which no market participants had even thought to attempt to price into the asset prior to their emergence.

Options, by their flexible nature, offer investors a way to protect themselves from these mispriced or unknown risks. In efficient markets, all knowable information should be reflected in the price of the underlying. However, there is much that cannot be known about events impacting the underlying far into the future. The volatility of the underlying can be thought of as uncertainty about what is known as well as everything that is unknown; asset prices in efficient markets only capture known information. Options contracts are constructs that allow volatility or what-is-unknown to be priced and traded. In this manner, options may be viewed as a form of insurance when used for hedging purposes. The writer of the option collects a premium for insuring a particular price for a particular transaction in an uncertain future, and the purchaser of the option holds a guarantee for that price and transaction, should future events transpire such that they require that guarantee as a means of protecting other positions or trades in their portfolio.

---

[97] Briefly: models typically rely on stochastic terms fitted via Monte Carlo simulation (in addition to parameters related to core features of the option such as the strike price, price of the underlying, and time until expiry). The "classical" model for options valuation is the Black-Scholes model, which relies fairly centrally on the assumption of a geometric Brownian motion as the model of the random walk of the underlying price, and the existence of a risk-neutral (martingale) measure. This model has been augmented since its creation with additional stochastic terms to capture further price behaviors such as jump diffusion (i.e. sharply discontinuous price gaps) and mean reversion in the price of the underlying. The interested reader may wish to refer to external resources for further detail on options pricing. Damodaran 2012, Chapter 5 gives a treatment of “Option Pricing Theory and Models”. Hilpisch 2015, Chapter 10 (Stochastics) gives an alternative computational approach to options valuation.

[98] One cost that these models for SmartPiggies tokens would need to account for is the capital cost of collateral, which is not present in traditional financial options and could increase the relative premia demanded for SmartPiggies tokens.

Example Real-World Use Cases

To make the understanding of why options might be useful more clear, a number of examples are presented to illustrate the potential benefits to counterparties in a variety of positions with respect to an option trade:

Call writer: A call writer is effectively selling insurance that pays out if the price of the underlying is above the strike price at expiry. Properly managed, call options writers are earning premiums by transferring financial risks to themselves.

Speculative call purchaser: A speculator in call options would typically purchase calls with a high strike price, which could be acquired quite cheaply as those options would be unlikely to pay off. At a low premium, such a speculator would benefit greatly from an extreme increase in the price of the underlying. Most likely however, the option will expire worthless. This position can be thought of as a form of leveraged expression of a directional view.

Hedging call purchaser: A call purchaser hedging another financial asset would typically have some kind of short position in that asset, such as a short futures contract or other commitment to purchase economic materials that are used to produce goods. Such individuals may want to protect themselves from extreme jumps in prices by purchasing call options that would pay off in case of extreme spikes in prices.

Portfolio hedge call purchaser: A portfolio manager in this position may have a net long position after a large run-up in the prices of certain assets in her portfolio, which should be partially sold off. If the investor is still bullish on the underlying in the long term, she may use call options to retain exposure to extreme upside while reducing the magnitude of her portfolio's overall directional position.

Put writer: A put writer is effectively selling insurance that pays out if the price of the underlying is below the strike price at expiry. As with call options, put writers are earning premiums by transferring financial risks to themselves.

Speculative put purchaser: A speculator in put options would typically purchase puts with a low strike price, which could be acquired quite cheaply as they would be unlikely to pay off. At a low premium, this speculator would benefit greatly from an extreme decrease in the price of the underlying. Most likely however, the option will expire worthless. As with speculative call options, this position can be thought of as a way for the speculator to express a leveraged directional view.

Hedging put purchaser: A purchaser of puts hedging another asset would typically have some kind of long position in that asset, such as physically holding the asset herself or producing economic goods directly (e.g. as a farmer or manufacturer). Such individuals may want to

protect themselves from extreme drops in prices by purchasing put options that would pay off in case of extreme drops in prices.

Portfolio hedge put purchaser: A portfolio manager in this position may have an overweight long position in her portfolio that she wishes to protect from extreme drops in price. She may purchase put options to establish a limit on potential losses in case of extreme drops in price.

Development of Options Markets

Access to options for risk management purposes has historically been limited to sophisticated investors and financial institutions who may negotiate directly with one another in order to create bespoke agreements to cover very specific risks. Due to the bespoke nature of these contracts, it is unlikely that they would trade on secondary markets as the interested counterparties would be difficult to identify. Furthermore, proper execution of these contracts requires highly trusted intermediaries, along with a strong regulatory environment to monitor those intermediaries. The intermediaries themselves require staffs of sophisticated professionals and sufficient infrastructure to process transactions and abide by regulations. For all of these reasons, contracts need to be of sufficiently large size to warrant the expense of writing and maintaining them. As such, they have been generally available only to banks, institutional players, and those with sufficient capital to participate at scale.

In terms of how they are traded, options fall in to one of two categories: over-the-counter (OTC) and exchange-traded. Until the most recent few decades, options were traded exclusively over-the-counter; that is, the options were negotiated between two counterparties for a specific deal. Starting in 1973 with the Chicago Board Options Exchange, standardized exchange-traded options have also been available to investors. Some key features and differences between these markets will be outlined in the sections below. In addition to the trading dimension, use of options may also be categorized by purpose. Typically, options are used for hedging, but may also be used for speculation. Options used for hedging may more often be OTC options, negotiated to hedge a specific risk asset position that one counterparty holds; in contrast, speculators may more often use exchange-traded options due to the relative ease of access to the market. In terms of market size, the over-the-counter market is many times larger than the market for exchange-traded options, reflecting the size and sophistication of the entities which are able to utilize OTC markets.[99]

Over-the-Counter Markets

Over-the-counter markets are, in a general sense, a form of peer-to-peer market. In an over-the-counter market, one counterparty (often desiring a hedge for another financial asset she has exposure to) negotiates directly with one or more counterparties to define the terms and price of a customized option contract which specifically suits her needs. Such markets require

[99] See https://www.bis.org/statistics/full_data_sets.htm for detailed data on the OTC and exchange-traded markets.

two-way communication channels to be active between the negotiating counterparties in order for the option to be written, and traditionally may be characterized as a relationship business due to the nature of this negotiation. Often, these OTC options are written on very large underlying notional values, for large premia which reflect the custom nature of the contract and the limited parties available to take the writer's side of the trade. Because financial agreements like options are only useful to the extent that there is a trusted regulatory and legal system to enforce settlement, the nature of the market requires formal business, banking, and legal relationships that are complex and expensive to maintain. As such, most individuals do not have access to the OTC market.

Options Exchanges

In recent decades, retail investors have been able to access options markets via options exchanges. In contrast to OTC markets, these exchanges list standardized options, which may be "fungibly" traded with one another.[100] To the extent that exchanges offer standardized products which offset positions that retail investors wish to hedge, these options may be used for a similar purpose as OTC options. Investors may find, however, that the standard lot sizes and expiries do not exactly line up to their offsetting positions and desired holding periods, so hedges will likely be approximate at best, assuming the underlying they wish to hedge is supported to begin with. For example, while Cboe offers options on over 3,500 single name stocks,[101] these equities are limited to US listed companies and are all physically settled (at predetermined lot sizes). If a retail investor wishes to purchase an option on a single name stock outside of the US, or would prefer cash settlement to physical settlement, Cboe will not be able to meet their needs for protection with standardized products. If another exchange does not offer the appropriate standardized option, without access to the OTC market, the investor may not be able to find protection at all.

Retail investors may also use these exchanges as leveraged betting markets, never intending to actually hold the underlying but simply using the facility of the exchange as a way to express directional views on the underlying. Though exchange-traded options may not be ideal for hedging purposes, they do more easily facilitate this sort of speculative trading. An example may help clarify how options can be used to provide leverage to speculators: assume an investor with $1,000 to trade has a belief that the price of AAPL will rise substantially from its current hypothetical level of $100 over the next month. She could purchase 10 shares today and hold them, hoping that her bet will pay off, or she could instead purchase 100 options for $10 each on AAPL with a strike price of $100 and a settlement date one month into the future. One month later, it turns out that our investor's hunch was correct: AAPL has risen to $120. Had she purchased the shares directly, she could now sell them into the market for $1,200, a 20% return on her investment. Had she purchased the options instead and sold them on the expiration date

[100] In a standardized option, the issuer will only write options on a specific set of underlyings, at predetermined lot sizes and for a limited range of strikes and expiries. This makes management of issuance easier, at the expense of flexibility for customers.
[101] A list of these companies is available in CSV format at:
http://www.cboe.com/trading-resources/symbol-directory/equity-index-leaps-options

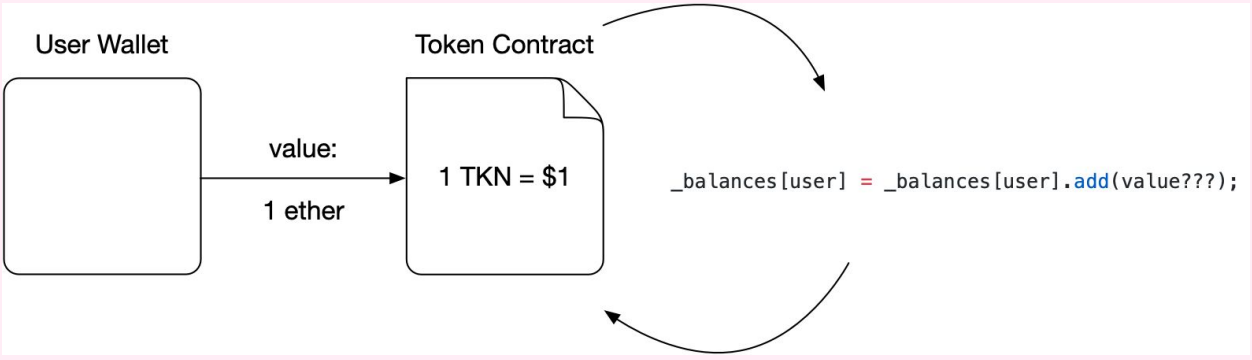
for $20 per option, she would stand to make $2,000, a 100% return. Of course, the downside risks are amplified as well; in the case where AAPL fell to $90 per share over the month, had our investor held the shares she would only suffer a 10% loss, whereas her options would have expired worthless and she would have lost 100% of her starting capital on the premia she paid.

Appendix II: An Illustrated Example of Oracles

The following example illustrates how oracle services which fetch external data in exchange for a fee may be implemented in practice.

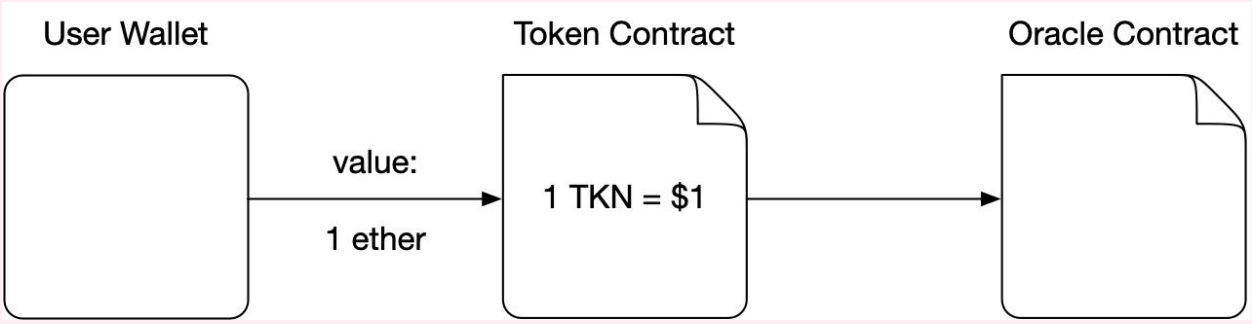
Consider a smart contract defining a token that is pegged to one U.S. dollar, which makes calculations based off of the current price of Ether (in US dollars) when a user initiates a transaction.

A user may deposit Ether into the contract in exchange for an equivalent amount of tokens in return, at the current exchange rate for USD/ETH:

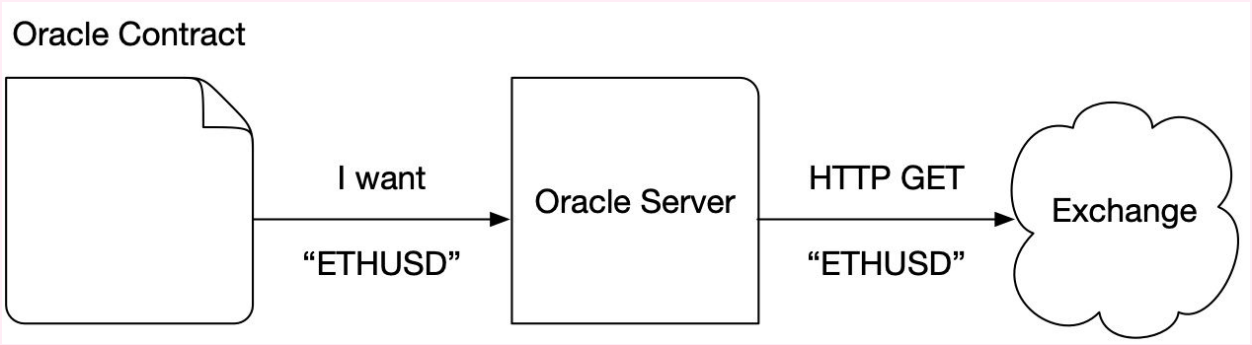


The contract would need to have access to the USD/ETH exchange rate when determining how many tokens to transfer, based upon the amount of Ether deposited.

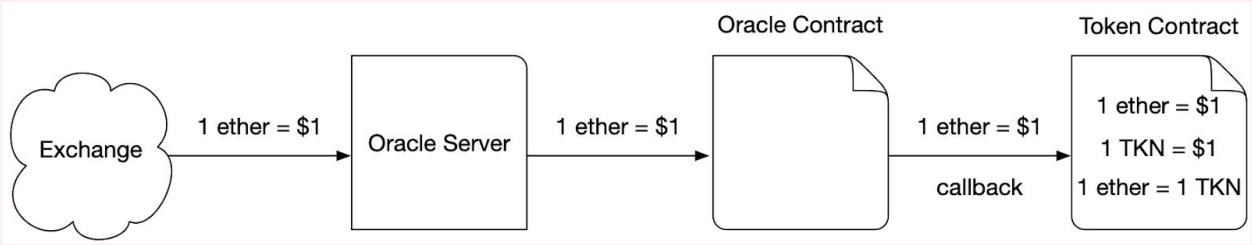
The smart contract could use an oracle to retrieve the price of Ether per US Dollar from an exchange API:



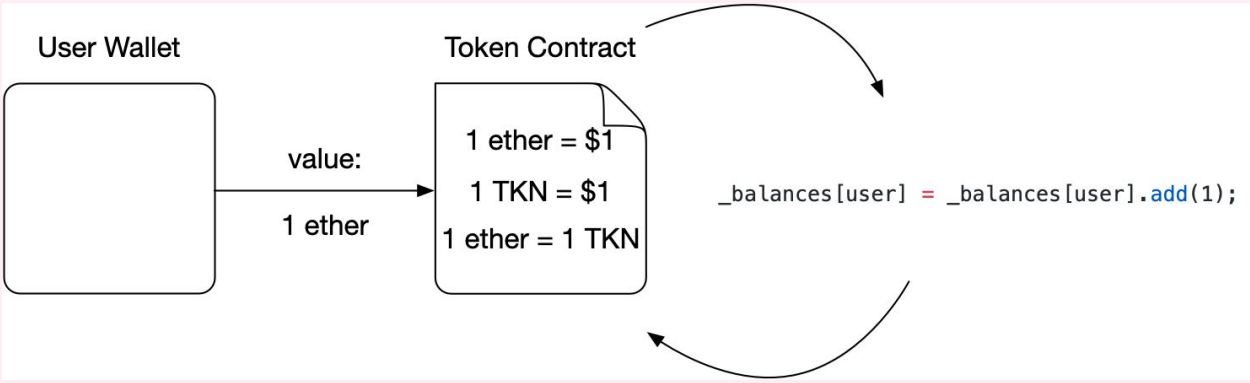
The oracle smart contract may then prompt the oracle server to retrieve the desired data from outside the Ethereum network:



The API will return data when it receives a request. In this simple case the exchange returns the price it has for the Ether to USD pair. The oracle server receives this returned data, and forwards it to the oracle smart contract. The oracle smart contract will use a callback to send the price data back to the token smart contract that initially asked for the data.



Now that the token smart contract knows what the exchange rate is for Ether to Dollars, the contract can calculate how many tokens to transfer to the user based upon the Ether deposited.



This is a trivialized explanation used to convey the general data flow when using an oracle or data provider to retrieve information that exists outside of a blockchain. Simplifications are made in the above description for the sake of clarity, at the expense of technical correctness. The “Risks and Mitigations” section of the paper details more technical implementations of oracles and data providers along with the risks these pose.