Using ABY to Construct a Decentralized Auction

©Luke Ahn, Stephen Bongner, Zen Yui, Rongxin Zhang
Crypto4Credit at Cornell Tech, New York City, NY
Published May 8, 2018

Abstract

The rise of Bitcoin and blockchains has risen in an interest in decentralizing all kinds of systems including DNS provision, payments, insurance, etc. In this project, we will explore the costs and benefits of decentralizing online auctions, emphasizing how they can be designed and implemented using robust cryptographic methods to maximize user security.

Background

Online auctions are run much like in-person, local auctions, but with added complexity and a layer of data collection not present in local auctions. In today's online auctions, users are required to register before they can sell or bid on an item and so the auctioneer can build a database of user feedback. Auctions generally close at a predetermined time but may be extended according to some known set of rules. There may be special rules set up like reserve prices to ensure that an item sells for some minimum perceived values. At an auction's close, the buyer and seller generally communicate electronically to arrange for payment and delivery of the goods (Montaldo).

Major sources of buyer concerns for the way auctions are run over the internet today include sellers who do not deliver the advertised goods, who fail to deliver in a timely manner, or who fail to disclose all the relevant information about the product or terms of the sale (Lee). Major sources of seller concerns include fraudulent chargebacks and poor feedback resulting from buyers not understanding the terms of the sale. Phishing is a problem on both sides (moviestarmaker). There are safeguards in place for some of these concerns. For example, as a means of mitigating seller fraud, some auctioneers function as escrow services that accept payment on behalf of the seller, then only release the funds after the buyer receives and approves the merchandise.

eBay and PayPal

eBay, founded in 1995, is the world's largest online auctioneer and a common household name. It is a publicly-traded company and in addition to hosting auctions acts as a trusted third party when there is a dispute between a buyer and a seller. Despite its "trusted" status, eBay has been scrutinized for a variety of security concerns that call into question its ability to objectively arbitrate disputes and even its ability to keep its buyers' and sellers' accounts and money safe.

eBay purchased the payment processor PayPal in 2002, a horizontal expansion that resulted in eBay holding control over every part of the auction process except the physical shipping of a seller's product. With control over the flow of money, eBay was uniquely positioned to be judge, jury, and executioner in all disputes between buyers and sellers. With no one "watching the watcher", eBay was thus incentivized to act in its own best interests. It made it very easy for buyers to report and obtain recompense for seller fraud, even when the seller is armed with proof that would have exonerated them in a federal courtroom (pictures, physical and digital documented and photographic records, etc.), which generated a new class of buyer fraud (Tims). eBay did separate itself from PayPal in 2014, spinning it out into a separate publicly-

traded company, but this was in response to it growing as a business separately and not due to concerns about dispute resolutions (Merced and Sorkin). Despite the separation, the problem is still prevalent, leading small and medium-sized sellers to abandon the platform for want of security (Levitin).

In 2014, eBay, or more specifically its account-holders, were victim to a network hack in which a database of user data was compromised. Compromised information included names, encrypted passwords, email and physical addresses, phone numbers, and dates of birth (Reisinger). Despite no credit cards numbers being stolen, the hackers had nonetheless stolen a trove of information that could be used for phishing and identity fraud. Phishing attacks against eBay and PayPal users saw a notable upsurge in the months following the compromise (Vatu).

PayPal, the payment processor, is not known to have been successfully compromised by any large-scale attack, but is nonetheless susceptible to small, targeted attacks. PayPal has in the past handed over account credentials to savvy phone-scammers claiming to be a user unable to log in and armed with a social security number or other personally identifying information (possibly obtained in the hack described in the preceding paragraph), and allowed these scammers to completely change account details to make it difficult for the original owner to recover their account and their funds (Smith).

The Problem with Trusted Third Parties

The section above highlights a real problem with online auctions today – even the most trusted of third parties are not reliable, whether intentionally, due to negligence of securing customer data, or both. Our project seeks to combat the problems with trusted third parties in online auction systems by offering a solution that doesn't require their participation. The rest of our time will be spent exploring the benefits and pitfalls of a decentralized auction system and a simple implementation demonstrating its capabilities.

Decentralized Auctions

The goal of a decentralized auction is to remove the need for a trusted third party, while maintaining and expanding the rights and privileges that buyers and sellers maintain through the current system. To name a few requirements:

1. The buyer must have some sense of security that the seller can't lie to the bidder about whether they are the current highest bidder so that they don't outbid themselves.
2. The buyer must be able to tell when they have won the auction.
3. The buyer should not be able to rescind payment once the seller has delivered the goods.
4. The seller must have some sense of security that the highest bid represents how much money they stand to make from the auction.
5. The seller must be able to collect payment and know whom to deliver their goods to.
6. The seller should not be able to obtain payment without delivering their goods.

We are building a decentralized auction that considers these security requirements and will explain in detail how they are met. Before we can discuss building a decentralized auction, we must consider some of the necessary building blocks we must utilize to make this dream a reality. The next section is spent discussing an important framework for secure multiparty computation and the cryptographic definitions and primitives it depends on to achieve this. Once that is understood, we will discuss implementation of a decentralized auction and the observations and conclusions we derived throughout the process.

ABY

ABY (Arithmetic, Boolean, and Yao sharing) is a framework for allowing mutually distrusting parties to evaluate a function on their private inputs without revealing anything but the function's output (Demmler, Scheider and Zohner). At a high level, ABY works like a virtual machine that abstracts from the underlying secure computation protocols and operates on data types of a given bit-length.

In ABY, Variables can be cleartext, as will be the case when one party knows the full value of the variable, or a data type called a share, which represents pieces of information that together have meaning but individually do not, or do not have enough meaning to have value. A simple example of a share is two 128-bit strings that when XORed together become a private key to an Ethereum account that holds joint funds. By themselves, the strings have no value, but together, they represent the key to a room full of money. The framework supports three kinds of sharing: arithmetic, Boolean, and Yao.

Sharing

Let us take a moment to discuss arithmetic, Boolean, and Yao sharing so that the reader understands what kind of information can be shared using the ABY framework.

Boolean Sharing

A Boolean share is a simple piece of binary information that can't be derived without the components to compute it. For example, consider a simplified version of the socialist millionaire's problem (Boudot, Schoenmakers and Traore). Two people, Alice and Bob, want to know if they're being paid fair wages. They want to determine if they make the same amount of money per hour for doing the same amount of work at the same level of quality, but neither wants to reveal how much money they make, not to each other nor do they mutually trust a third party to hold this information. They at least trust each other not to lie, though. They get a bunch of small boxes together with a thin hole cut into them to slip messages into and label them with the amount of money per hour they might make ($10, $11, $12 per hour, etc.) and lock them all with keyed padlocks. Now Bob throws away all the keys except for the one that will open the box that corresponds to how much money he makes. He passes the boxes to Alice, who puts a slip of paper in each box. Each slip has the simple message "no" except for the one that goes into the box that corresponds to how much money Alice makes. Now Alice returns the boxes to Bob, who uses his key to open the only box he can to retrieve the slip of paper inside. He reveals the paper to Alice. If the paper says "yes" on it, they both make the same amount of money. If it doesn't, they don't, but neither knows exactly how much the other makes or even who makes more. The Boolean share was "do Alice and Bob make the same amount of money?" and the individual components, how much money Alice and Bob individually make, do not individually hold enough value to compute the value of the share.

Arithmetic Sharing

An arithmetic share is a piece of information that must be obtained mathematically that requires a certain set of inputs before it can be computed. Consider an example in which Alice, Bob, and Charlie need to know the Euclidean distance between two points. Alice holds the X coordinate of both points, Bob holds the Ys, and Charlie holds the Zs. None want to share their coordinates with the others. They each input their coordinates into a computer, which then broadcasts the Euclidean distance between the points in 3-dimensional space created by putting the coordinates together. The share is the distance between the two points and the individual components each lack two of the dimensions required to make the points the three collaborators need the distance between.

Yao Sharing

Yao sharing, or a garbled circuit, is a cryptographic protocol that enables secure computation in which two mistrusting parties can jointly evaluate a function over their private inputs without the presence of a trusted third party (Wikipedia Foundation). The function is a Boolean circuit, generally a collection of AND and XOR gates that takes inputs and puts them through the gates to obtain a final output. Consider the locked box socialist millionaire problem above again. This problem could be represented by a simple circuit with only AND gates. Alice and Bob both make an amount of money per hour that can be represented by a 4-bit string. Alice feeds her salary in as 1011 and Bob feeds his salary in as 1010. Here's what the circuit would look like:

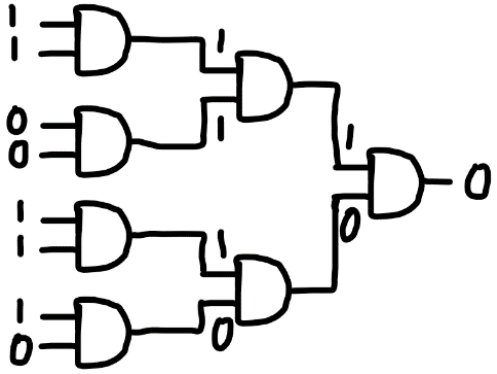


Figure 1 – Socialist Millionaire Problem Circuit

A garbled circuit requires more than simply specifying a Boolean circuit to take the inputs through. The Boolean circuit is just one component of a protocol, which must keep Alice and Bob from seeing each other's inputs. The protocol that succeeds in this goal requires the following steps:

1. A Boolean circuit is generated and known by all parties. Ideally, the circuit is optimized for performance. The circuit drawn above is far from optimized. Truth tables are built from the circuit.
2. The circuit is garbled or encrypted by one party called the garbler. Each wire of the circuit receives two randomly generated *k*-bit security parameters each representing 0 or 1. The values in the truth tables are replaced with the labels. The output entries of the truth table are then encrypted with the corresponding input labels to create a garbled table. The garbled table is randomly permuted, so the output value can't be determined from the row.
3. The garbler gives the garbled circuit and their encrypted input (the labels) to the other party, called the evaluator. The evaluator learns nothing about the garbler's input because the labels are random.
4. The evaluator needs the encryptions for their inputs. Through a process called a 1-out-of-2 oblivious transfer, the evaluator receives from the garbler the encryptions of their inputs. The idea behind a 1-out-of-2 oblivious transfer is that the evaluator gets to ask for exactly one element (e.g. X_0^b, the label for $b = 0$, out of the possible two elements X_0^b and X_1^b, without the garbler learning which element was queried and without the evaluator learning about the other element not retrieved.
5. The evaluator goes through the circuit, finding they can decrypt exactly one row of each truth table (the decrypted row is itself an encrypted label, so the evaluator learns nothing of its actual value), until eventually obtaining the label representing the output.
6. Either the evaluator gives the label to the garbler who can decrypt it, or the garbler shares the key required to decrypt the output so they can decrypt it, and one or both of them learn the output.

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

→

a	b	c
X_0^a	X_0^b	X_0^c
X_0^a	X_1^b	X_0^c
X_1^a	X_0^b	X_0^c
X_1^a	X_1^b	X_1^c

→

Garbled Table
$Enc_{X_0^a, X_0^b}(X_0^c)$
$Enc_{X_0^a, X_1^b}(X_0^c)$
$Enc_{X_1^a, X_0^b}(X_0^c)$
$Enc_{X_1^a, X_1^b}(X_1^c)$

Figure 2 – Creation of a Garbled Table

Security Guarantee

The most important guarantee that the ABY framework makes is that no party can learn the other party's inputs to the protocol. The sender encrypts all possible inputs (s_0, s_1) and the receiver inputs a single bit $c \in \{0,1\}$ and obtains s_c as output, and this is done in a way that the sender does not learn any information about c (the receiver never learns s_{1-c} either).

How does the sender not learn anything about the bit c that the sender provides to obtain the encryption of their input to the ABY protocol? This question is answered by the use of oblivious transfer, and more specifically, 1-out-of-2 oblivious transfer (1-2-OT). Consider the following protocol that makes use of a machine called an oblivious transfer machine (OT machine). Whenever the OT machine receives a bit from the sender, it flips a coin. On heads, it sends the bit to the receiver, but on tails, it sends '#' to the receiver to indicate that it received a bit from the sender, but it's not sharing that bit. (Ostrovsky)

1. The sender inputs a large string of random bits s into the OT machine, which relays roughly half of them to the receiver and relays '#' in place of the other half.
2. The receiver creates two equal-sized sets of indices of bits from s. One set is completely put together at random. The other set consists only of indices for which the receiver obtained the corresponding bits of s and not '#'. This other set will correspond to the bit the receiver wants to receive from the sender. The order of the sets sent to the sender are randomized but remembered by the receiver.
3. The sender chooses an index from the sets provided by the receiver. The sender chooses two bits to send to the receiver. The sender XORs the first bit with the value at the chosen index in the first set and XORs the second bit with the value at the chosen index in the second set. Then the sender outputs both results to the receiver.
4. The receiver XORs the bit with the value in the appropriate set to obtain the bit they desire.

For example, the sender provides 001010 to the OT machine. The OT machine relays #01##0 to the receiver. The receiver creates a set of indices at random, say {2, 5} and a set of random indices from the bits that weren't '#'s, say {6, 1}. The receiver gives both sets to the sender. The sender chooses index 1 and bits 0 and 1. The sender sends 0 = 0 XOR 0 (the second bit of 001010 XORed with 0) and 1 = 0 XOR 1 (the sixth bit of #01##0 XORed with 1) to the receiver. The receiver XORs the second bit received (1) with 0 (the sixth bit of #01##0) to obtain the desired message (1).

When this protocol is used, the sender won't have any idea which bit the receiver wanted, but the receiver gets what they wanted anyway! It's worth noting that the protocol described here depends on the sender and receiver both being honest, but it can be modified in such a way to provide security when a sender is malicious (i.e., the sender wants to learn the bit they are giving the receiver, and by extension the receiver's input to the ABY protocol).

Building a Decentralized Auction

The ABY framework's example code contains implementations of several simple examples, including an extension of the millionaire problem in which Bob and Alice wish to know who has more money without revealing how much either of them has. We determined that it is possible to extend the millionaire problem code to create the foundation for a decentralized auction.

Our implementation requires solving the millionaire's problem, which is like the socialist millionaire's problem, except that Bob and Alice want to know who has more money, not just whether they have the same amount of money. This is accomplished via the protocol proposed by Loannidis and Grama, which runs efficiently (in polynomial time) and has been proven to be secure by its use of oblivious transfer (Ioannidis and Grama). Next, this implementation of the millionaire's problem must be performed by multiple parties.

To do so, we need to solve several issues:

1. Currently, the ABY framework only supports a two-party computation. However, in a real auction, there will likely be more than two parties. Therefore, we need to be able to perform Yao's Garble in an efficient manner between more than 2 parties.
2. Buyers must commit their money before the auction begins such that they cannot bid more than the amount of money they have, or bid without any money.
3. Buyers should not be able to gain a competitive advantage over other buyers simply by viewing the amount deposited into the transaction.
4. Buyers should not be able to change their bids once the winner has been revealed.
5. Buyers must not be able to hold up a round by not submitting an input.
6. There must be a possibility to dispute the winner.

We will provide solutions to each of the above issues sequentially:

Issue 1

Performing Yao's Garble between more than 2 parties is as simple as running the Garble pairwise against every participant in the auction. The ABY framework provides the mechanism for running against one pair, so we simply create a method to run it against every pair and report whether the caller has bid more money than any of their competitors. If the caller has bid less money than any one competitor, they are not the highest bidder and the method returns false. If the caller has bid more money than all other competitors, they are the highest bidder and the method returns true for all pairwise comparisons. This caller will then know they can submit the win and lay claim to the prize.

Issue 2

To prevent buyers from bidding more money than they have bidding without any money, users must send their money to an escrow service that is controlled through a third party. In our case, we use a Smart Contract which acts as a trust-less escrow agent. It is standard practice in real world auctions to provide a deposit or proof of wealth (Christie's). There are economic penalties like taking away a user's money if

it is discovered that they won while bidding more than they actual deposited. We perform this with a simple function in which the winner's bid is checked against their deposit. If it is more than what they proposed, then they lose their all or a portion of their deposit.

Issue 3

Since all money sent to the blockchain is public, buyers who deposit more money have a natural advantage over those that deposit less. For instance, if Alice deposits 100 while Bob deposits 50, Alice will know that if she bids 51, she is guaranteed to win. This issue extends to situations when there are more than 2 buyers. A simple solution is to have all buyers deposit the exact same amount of money. For instance, in an auction all users must deposit 200 to join the auction. As a result, no one can gain any advantage as they don't know how money each buyer is willing to bid. However, this limits the maximum amount that can be bid in the auction. We can solve this by having multiple auction rounds for the same item in which bidders that bid the max amount of the deposit are transferred to a second auction with a higher buy-in. The bidders continue up the rounds until a winner is found.

Issue 4

We can prevent bad buyers from changing their bid amount after the winner has revealed their bid by forcing buyers to commit their bid before the auction begins. Each user signs a message that contains:

$$pk_i||bid_i||nonce_i \text{ where } i = \text{buyer}$$

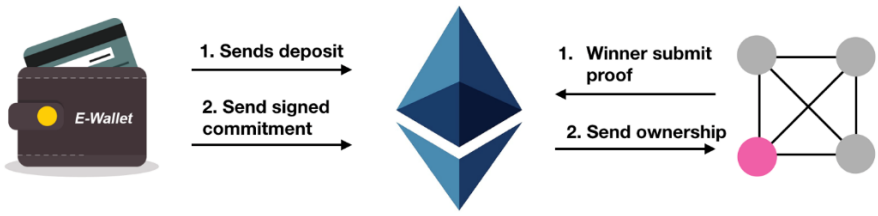
Each user is required to submit one and only one commitment message. As a result, each buyer commits to a bid before the bid begins. To prove that a bid is valid, the winner must present their bid in plaintext along with the nonce to verify that the they did indeed bid this amount. Our smart contract has this functionality built in to it.

Issue 5

To prevent bad buyer from holding up an auction if they are the winner, there is a create time limit for submitting a proof of winning. If no winner submits a winning bid value in time, then all participants must reveal their bid and nonce, and prove that they created the commitment. In this case, the person that does not reveal their bid or the highest bidder will have their money "burned" to a 0x000 address, while the rest of the money returned to the bidders.

Issue 6

If a buyer believes that the winner is wrong, i.e., they bid higher than the winner, they can submit their bid and nonce to the contract to dispute the winning claim. The check is simple: as long as the buyer that is disputing can prove that a commitment message is created from a higher bid value than the winner's, the disputer becomes the new winner and the person that pretended to have more money receives a punishment fee for submitting a false value. However, this does not prevent an adversary from submitting less money into the ABY comparison function that is not the value they submitted in their commitment value. As a result, the adversary can always challenge the winner, take the win, and invalidate the previous winner. We rely that whatever people submit into the ABY framework is semi-truthful, i.e., that a user cannot submit lower than their commitment value.



Step 1: Set up

a) Buyer sends specified amount of money e.g $200
b) Each buyer sends commitment of: bid || nonce || public key

Step 2: Compute winner using ABY

a) Buyers know they are winner or not.
b) Winner submits in 1 hour: bid || nonce || public key
c) If there is no dispute, then money is transferred in 1 hour

Figure 3 – Auction Result Smart Contract

The following section contains the pseudocode for the smart contract:

```
Variables
deposit
commitment
winner
dispute_timer
set_val

Deposit()
If sender.value == set_val
      deposit[sender.address] = sender.value

Winner(nonce, bid_val)
If deposit[sender.address]
If verification(sender.address || nonce || bid_val) ==
commitment[sender.address]
      winner[sender.address] = bid_val
      dispute_timer
else:
      // punishment

Dispute(nonce, bid_val)
If dispute_timer - current_time < 1 hours:
      If verification(sender.address || nonce || bid_val) ==
commitment[sender.address]:
            if winner[sender.address] < bid_val:
                  if winner[sender.address] != sender.address:
                        winner[]
else:
      # punishment for all other scenarios
```

References

Boudot, Fabrice, Berry Schoenmakers and Jaques Traore. *A Fair and Efficient Solution to the Socialist Millionaires' Problem*. 15 April 2003. 2 May 2018. <https://www.win.tue.nl/~berry/papers/dam.pdf>.

Christie's. *Buying at Christie's*. n.d. 8 May 2018. <https://www.christies.com/buying-services/buying-guide/register-and-bid>.

Demmler, Danel, Thomas Scheider and Michael Zohner. *ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation*. 9 June 2015. 2 May 2018. <http://thomaschneider.de/papers/DSZ15.pdf>.

Ioannidis, Ioannis and Ananth Grama. *An Efficient Protocol for Yao's Millionaires' Problem*. 6 January 2013. 7 May 2018. <https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2003-39.pdf>.

Lee, Joel. *10 eBay Scams to Be Aware Of*. 28 June 2017. 30 April 2018. <https://www.makeuseof.com/tag/5-ebay-scams-to-be-aware-of/>.

Levitin, Victor. *Why We Almost Abandoned Selling on eBay, And What Made Us Stay*. 21 October 2015. 30 April 2018. <https://crazylister.com/blog/selling-on-ebay-business/>.

Merced, Michael J. De La and Andrew Ross Sorkin. *EBay Does About-Face in Spinoff of PayPal Backed by Icahn*. 30 September 2014. 30 April 2018. <https://dealbook.nytimes.com/2014/09/30/ebay-to-spin-off-paypal-adopting-strategy-backed-by-icahn/?_php=true&_type=blogs&_r=0>.

Montaldo, Donna L. *How Online Auctions Work*. 18 December 2017. 30 April 2018. <https://www.thebalanceeveryday.com/how-online-auctions-work-940780>.

moviestarmaker. *SCAMS committed by eBay buyers*. 18 February 2012. 30 April 2018. <https://community.ebay.com/t5/Archive-The-Front-Porch/SCAMS-committed-by-eBay-buyers/td-p/2662626?rmvSB=true>.

Ostrovsky, Rafail. *Oblivious Transfer*. March 2005. 5 May 2018. <http://web.cs.ucla.edu/~rafail/TEACHING/WINTER-2005/L10/L10.pdf>.

Reisinger, Don. *eBay hacked, requests all users change passwords*. 21 May 2014. 30 April 2018. <https://www.cnet.com/news/ebay-hacked-requests-all-users-change-passwords/>.

Smith, Chris. *Your PayPal account can be hacked more quickly than you think*. 4 January 2016. 30 April 2018. <https://bgr.com/2016/01/04/paypal-account-security-hackers/>.

Tims, Anna. *It's seller beware as eBay's buyer guarantee is exploited by scammers*. 25 April 2016. 30 April 2018. <https://www.theguardian.com/money/2016/apr/25/ebay-seller-beware-buyer-guarantee-exploited-scammers>.

Vatu, Gabriela. *"Recent Activity" Phishing Attacks on PayPal, Due to eBay Hack?* 22 May 2014. 30 April 2018. <https://news.softpedia.com/news/Recent-Activity-Phishing-Attacks-on-PayPal-Due-to-eBay-Hack-443369.shtml>.

Wikipedia Foundation. *Garbled Circuit*. n.d. 2 May 2018. <https://en.wikipedia.org/wiki/Garbled_circuit>.