THE AUTHAUCTION SECURE AUCTION SYSTEM

Michael Rosenberg Dr. Dennis Shasha

1 Abstract

In this paper we present a cryptosystem that will allow for fair sealed first-price auctions to be conducted over the internet without the need for a trusted third party. The cryptosystem consists of a set of protocols defined between clients and a server that utilize cryptographic primitives to ensure the infeasibility of cheating the system without detection. The server software is simple, as it is required to perform no cryptographic computations. The suite is compared with and contrasted against the VEX[1] protocol for auctioning in online ad exchanges.

2 Introduction

It is fairly common for particularly wealthy families to host estate auctions following the passing of a family member. This process would normally entail a very steeply-priced lawyer acting as an arbitrator over the entire process to ensure that no party is unfairly treated. The current state of affairs could be greatly improved using an easily deployable bundle of software. AUTHAUCTION provides just this. It is a simple, if not rudimentary, and secure suite with which such families may conduct sealed first-price auctions without human supervision. This example of wealthy families is only one use case. Any situation in which the items being auctioned and their prices are to be kept secret from outsiders would also lend itself to the use of this system.

3 Terminology

We will define several terms so that they may be used in this paper without further ambiguity. The Server is the central location and computer where information about Estates is stored and where auctions are conducted. We define an Estate as a series of Auctions on individual items held among some group of people (called members). The Phases of bidding are the steps each client must take in the bidding process (our system requires two Phases). Corresponding to each Server (the client software supports participation in multiple Servers) there is a Global Administrator. To each Estate there is an Estate Administrator as well as the members of the Estate.

4 Specification

Our cryptosystem relies upon the following primitives:

Message Authentication Code Function	$\text{MAC}_k(\text{msg})$
Hash Function	HASH
Public-Private Key Digital Signature Algorithm	$\text{SIGN}_k(\text{msg})$, $\text{VER}_k(\text{msg}, \text{sig})$
Cryptographically Secure Pseudorandom Number Generator	CSPRNG

The basic strategy in AUTHAUCTION to externalize all cryptographic operations to the users themselves as opposed to the server. The security of the system relies on having every user use client-side software. The advantage is that the clients don't need to trust the server, however the disadvantage is that the clients need to put more effort into the entire process than they would when using Ebay, for example

[1] S. Angel and M. Walfish. *Verifiable Auctions for Online Ad Exchanges*. `https://dl.acm.org/citation.cfm?id=2486038`

4.1 Setup

The Global Administrator (the owner of the auction server) creates a new Estate and a new Estate Administrator. The Estate admin then proceeds to add users to the estate by username. Every user is expected to generate a new keypair corresponding to the particular Estate and disseminate his/her respective public key to all other members of the Estate, preferably in a secure, out-of-band, manner, e.g., by reading over the telephone. The Server is not expected to know of any information about the users other than their usernames. One may propose that the Server have a list of the members' public keys for the sake of ease-of-access but this practice is likely to breed trust in the Server which can be severely detrimental to the system.

4.2 Auction Process

The Estate Admin predetermines the list of auctions. The list is comprised of each item's name, description, and an ID that is randomly generated by the Estate Admin. Each auction ID must be unique within the Estate. If it is not the case, the client software will report a warning to the user upon Estate entry. The user who receives this error is to report it immediately to the Estate Administrator as well as all other members of the Estate, requesting a new item list with unique item IDs.

Bidding in an auction is broken up into two phases. Phase #1 is for the placing of bids in an irrevocable manner. The data submitted here are called bid commitments. Phase #2 is for the submission of the data both revealing the bid and authenticating the commitment.

During Phase #1, each member of the Estate must first generate a new random string of bytes p heretofore known as the Prekey. The Prekey and the auction item's name, description, and ID are all hashed, concatenated and hashed again to produce the key to be used with the MAC function:

$$k = \textsc{hash}(\textsc{hash}(\text{name}) \parallel \textsc{hash}(\text{desc}) \parallel \textsc{hash}(\text{id}) \parallel \textsc{hash}(p)).$$

The user then computes and submits $s = \textsc{sign}_i(\textsc{mac}_k(b))$ where i is the user's private key and b is the user's bid.

The Server institutes a waiting period following the end of Phase #1 that allows every member of the Estate to download the bid commitment information of every other member. Each member inputs this block of information into his/her accompanying AuthAuction client-side software for local safekeeping. The Server is to wait until each member has acknowledged that the commitment data has been saved before moving on to Phase #2.

During Phase #2 the information necessary to authenticate the bid commitment data entered in Phase #1 is submitted. The user simply submits b and p to the Server. With this information and the public key of the bidder, any other member of the Estate can reconstruct k as above and recompute $z = \textsc{mac}_k(b)$.
so as to assert that $\textsc{ver}_u(z, s)$ succeeds, where u is the public key of the user whose bid is being verified.

If at any point in the protocol one user fails to submit valid data or submit data at all, every other member of the Estate must wait until the user complies. Furthermore, if any member announces that something is not correct, the auction is to be restarted. This is due to the fact that it is impossible to determine the perpetrator of the incident and it may sometimes be impossible to even determine if the whistle-blower is telling the truth. Thus the only way to ensure an untampered auction is to complete one without any user complaints. Fortunately, each user has an incentive both to behave honestly and to check on others.

5 Analysis

5.1 Assumptions

To formalize AuthAuction's correctness, we must first formalize the assumptions that we make about the underlying system:

(1) All cryptographic primitives are sufficiently secure for their appropriate use.

(2) No private key is known to anyone other than the owner of the key.

(3) All keys and random strings are sufficiently long for their appropriate secure use.

5.2 Guarantees

AuthAuction makes a number of specific guarantees that relate to user privacy and integrity:

(1) No user can retrieve bid amounts prematurely, i.e., before the beginning of Phase #2.

(2) No user can tamper with bid commitments after the end of Phase #1.

(3) The server cannot ignore bids from any subset of Estate members.

(4) No user may deny their bid commitment after submission.

(5) No user can forge bids or bid commitment data.

5.3 Reasoning

The purpose of this cryptosystem is to ensure that no party under any circumstance should have an unfair advantage over the other participants. Any and all attempts at such an advantage are detectable but not necessarily traceable to the perpetrator. The following is the reasoning that has gone into the design of the cryptosystem.

We begin with the use of the MAC function. Concretely, a MAC, or message authentication code, function is a one-way function that computes the digest of an input of arbitrary length given a fixed-length key. It ensures that, given $\text{MAC}_k(s)$, someone without the key k is unable to calculate $\text{MAC}_k(s')$ for any $s' \neq s$. A MAC function is often used for proving and authenticating the integrity of data. The key used in our MAC function is a mix of random data and the auctioned item's metadata. The reason for the use of the item's name and description in the construction in the key is to ensure that the bid is valid only for that particular item. This assumes that each item will have a different name and/or description. To further secure this, we also include a randomly generated item ID in case a name and description are repeated for any reason. Furthermore, the construction $\text{HASH}(\text{HASH}(*)\|\text{HASH}(*)\ \ldots)$ is used as opposed to $\text{HASH}(*\|*\|\ \ldots)$ in order to prevent potential collisions. If a suffix of the item name were removed and prepended to the item description, the value of (name $\|$ desc) would remain the same, and thus the digest of it would remain the same as well. This occurrence is avoided if the former hashing construction is used.

We choose to use a MAC as opposed to a hash because the signature function is not intended to obfuscate the underlying data. Knowing this, an attacker could theoretically extract the hash from the signature $h = \text{HASH}(b)$ (again, where b is the user's bid). Now deriving b given h is simple. Simply iterate d over all possible values of b until $h = \text{HASH}(d)$ is satisfied. Since there are not many possible values for b (in terms of money, around 10^9), this attack is quite feasible. The use of a random value to frustrate such a brute-force attack would be necessary. One solution would be to concatenate the bid value to a random secret and evaluate the hash of that. However, this approach is vulnerable to a length-extension attack for Merkle–Damgård hash functions. This is avoided with the use of a suitable MAC function, such as HMAC. Note that, for sufficiently long random values use in the key, it is computationally infeasible to derive the value of the bid from the MAC under assumptions 1 and 3.

The choice between using a MAC vs. a symmetric cipher for committing to a datum is, to an extent, arbitrary. However, the use of a MAC has a few advantages[2]. First, the key size is not bounded. If it exceeds the block size of the underlying hashing algorithm, it is hashed and the result is used as the key. Second, the block size can be relatively easily changed by swapping out the underlying hashing algorithm. As an example of the flexibility this allows us to have, note that SHA2, offers more choices in block size than AES (the former coming in flavors of 224, 256, 384, and 512 bit block sizes, while the latter comes with the choice of a 128 or 256 bit block size).

The need for signatures in the system is much more obvious. Without digital signatures, it would be trivial for an attacker to masquerade as another member of the Estate.

The two-Phase commitment system is intentionally very similar to a blinding protocol, the only differences being that it is authenticated, it uses a MAC instead of an XOR, and the actual plaintext data is submitted in the second step. The logic is that all members of an Estate will have access to each other's bid commitments after Phase #1. At this point, everyone's bids are immutable, because if they are mutated later in Phase #2, they will

[2] http://crypto.stackexchange.com/questions/12247/what-are-the-pros-cons-of-using-symmetric-crypto-vs-hash-in-a-commitment-scheme

no longer pass the verification step when verified with the commitment data given previously. In this way, every user may now submit their bid without the possibility that another user will see it and intentionally out-bid him.

The reason all users must wait for a non-responsive user to comply is that it is inherently impossible to determine the cause of the error (whether it be a lack of response or submission of invalid data). From the perspective of any other user, it may well be the Server itself that is ignoring or modifying data.

5.4 Correctness

Here we show that all guarantees are satisfied as stated above.

No user can retrieve bid amounts prematurely, i.e., before the beginning of Phase #2

The only information about a bid that any member of an Estate other than the bidder has access to is $\text{MAC}_k(b)$ where k is the appropriate key and b is the bid amount. Retrieving b from this is equivalent, if not more difficult, than retrieving a known input to a MAC with an unknown key. Based on assumption 1, we assert that the cryptosystem is secure against this threat.

No user can tamper with bid commitments after the end of Phase #1

Because of the Server-instituted waiting period between phases, an attempt to tamper with bidding information on the Server at any point after the clients had already downloaded the information would be completely ineffectual. Tampering with bidding information on the Server before all members have downloaded the bidding information (recall that Phase #2 does not commence until all members have acknowledged that they have downloaded the bidding information) is possible but could not result in any member's benefit. If a member were to tamper with his own commitment data on the Server (implying that the member has access to the private key used to sign the commitment), the member receives no unfair advantage, as this (albeit unscrupulous) action is equivalent to the member making his initial bid commitment. If, on the other hand, one were to tamper with another member's commitment data before any other member has the ability to download it from the Server, it would show after Phase #2 that that member's bid does not match the commitment data provided. Since the standard practice for an event like this is to repeat the auction (with a different ID), no member has gained an unfair advantage. The only way an attacker would be able to hide the fact that a bid commitment was tampered with would be if the attacker were able forge a valid commitment with the private key of the target user, this is in violation of guarantee 5. Furthermore, if the attacker were to strike after a subset of members had downloaded the commitment data but before a different, non-empty subset of members did the same, the members of the estate could convene and determine that the commitment data was tampered with. Though the act of convening is not necessarily a feature of the system, it is logical since we are operating under assumption 2.

It is also a possibility that the Server can unilaterally skip the waiting period (whereby all members of the Estate download the commitment data and acknowledge that they have done so) after Phase #1 and move on to Phase #2. The onus is upon the members in this case to blow the whistle on the Server and demand the data before continuing.

The server cannot ignore bids from any subset of Estate members

When commitment data is imported into the client software, a cursory check is performed to ensure that every member's commitment data has been submitted. If this is not the case, the member is presented with a visible warning of just that. It is then, again, the responsibility of the member to inform the other members as well as the Server that there is missing data.

No user may deny their bid commitment after submission

Every bid commitment is signed with the corresponding member's private key. This commitment can be verified

given the member's public key which was distributed at the setup of the Estate. Commitment denial is defined as the assertion that a signed commitment that has been verified was not signed by the member in question. This claim is equivalent to the claim that, given a signed message $s = \text{SIGN}_i(z)$ and a public key u_1, one can generate a new public key u_2 such that $\text{VER}_{u_1}(z, s)$ and $\text{VER}_{u_2}(z, s)$ both succeed. Under assumption 1, this is computationally infeasible.

No user can forge bids or bid commitment data

[For brevity, we define $\text{G}(\text{name}, \text{desc}, \text{id}, p, b) = \text{MAC}_{\text{HASH}(\text{HASH}(\text{name}) \,\|\, \text{HASH}(\text{desc}) \,\|\, \text{HASH}(\text{id}) \,\|\, \text{HASH}(p))}(b)$.] There are multiple ways a user could go about forging another user's data. Recall that, under assumption 3, no user has access to any other user's private key. Suppose, firstly, that Mallory (the would-be forger) has managed to circumvent the login system, allowing him to login as another member, Alice. If the current auction is in Phase #1, Mallory may submit his own commitment data, $c = \text{SIGN}_m(\text{G}(\text{name}, \text{desc}, \text{id}, p_1, b_1))$, for some private key m, to the Server without Alice's knowledge or consent. Suppose this action was unnoticed by both the Server and Alice. The problem remains that regardless of what Mallory submitted, it could not have possibly been signed with Alice's private key. By Phase #2, Mallory will be forced to enter p_2 and b_2 (the Prekey and bid amount, respectively). Note that these values do not have to be the same as the values chosen for the commitment generation. Passing verification would require Mallory to solve for m, p_1, p_2, b_1, b_2 such that c is a valid signature of $\text{SIGN}_m(\text{G}(\text{name}, \text{desc}, \text{id}, p_2, b_2)$ under Alice's key a. This should not be feasible under assumptions 1 and 3. Furthermore it would be even more difficult for Mallory if he were to fix a particular b_2 as Alice's public perceived bid amount.

It may also be possible that Mallory submits only his own data in Phase #2. In this case, Mallory would be forced to solve for b_2 and p_2 such that Alice's commitment, $A = \text{SIGN}_a(\text{G}(\text{name}, \text{desc}, \text{id}, p_1, b_1))$, is a valid signature of $\text{SIGN}_m(\text{G}(\text{name}, \text{desc}, \text{id}, p_2, b_2)$. This should be similarly infeasible.

Mallory also has the ability to copy Alice's bidding information from previous auctions into a new auction. That is to say that the p and b are known to Mallory before the attack even begins. This is known as a replay attack and it is prevented with the use of unique item names and descriptions as well as unique item IDs for every auction. If IDs and item name and description were equal, that is, when $\text{id}_1 = \text{id}_2$, $\text{name}_1 = \text{name}_2$, and $\text{desc}_1 = \text{desc}_2$, then the solution to $\text{SIGN}_m(\text{G}(\text{name}_1, \text{desc}_1, \text{id}_1, p_1, b_1) = \text{SIGN}_m(\text{G}(\text{name}_2, \text{desc}_2, \text{id}_2, p_2, b_2)$ is trivial: $p_1 = p_2$ and $b_1 = b_2$. Solving this problem when all three values are unique, however, is infeasible.

6 Comparison

The two-part bidding process used here is very similar to that of VEX insofar as they both feature a commitment stage and a revealing stage. The essential difference is how the bids are revealed and to whom. VEX dictates that every user should share his bid amount (more specifically, *decommitment data*, from which the bid amount can be trivially derived) with the auctioneer and that the auctioneer should keep the bid amount secret from the rest of the bidders. AUTHAUCTION states that the Server is to collect all of the bid amounts and display them for every user to see. We choose to not support private integer comparison as it would require a trust in the Server that can fairly easily be violated, i.e., the trust that the Server keeps all bid amounts secret and only releases the necessary information to each member of an Estate to prove that their bid was less than the winning bid. This level of trust did not appear to the creators to be worth the risk and so was intentionally left out. The alternative for private integer comparison is through an interactive protocol. This, too, was not included in the specification due to the inherent complexity of performing a peer-to-peer interactive protocol over the internet with no additional provided infrastructure.

VEX happens to take a quite different approach to bidder deniability. It is explicitly stated that there is nothing linking a bid with the bidder. AUTHAUCTION employs the use of a signature algorithm to ensure that each bid is strongly tied to its respective bidder. The practical difference between these two approaches manifests itself in the feasibility of a forgery attack or tampering with data. AUTHAUCTION asserts that a successful forgery would be infeasible under assumption 1. This also applies to tampering. The benefit that plausible bid deniability affords the

users of the VEX cryptosystem is privacy of bid amount. This is to say that even if a user obtained all bid amounts, there would be nothing linking them to the bidders themselves. This particular facet of privacy is one that took precedence in the creation of VEX and is not supported by AUTHAUCTION.

Because hash chains are used to represent the monetary bid amounts in VEX, scaling for increased precision (such as allowing the user to specify cents or fractions of cents) increases the work performed in the hash chain exponentially. For example, a $100 bid commitment in an auction that only supports dollar values and nothing less would be represented as $H^{100}(s)$ where s is some random seed. If this auction supported bids of dollars and cents, the same $100.00 bid commitment would be represented as $H^{10000}(s)$. The commitment scheme used by AUTHAUCTION scales linearly with the input length since the bid commitment is represented as a MAC of a string which, in turn, represents the bid amount. More concretely, the difference between a $100 bid when cent values are unsupported and when they are is $\text{MAC}_k(\text{“100”})$ versus $\text{MAC}_k(\text{“10000”})$.

VEX's use of an interactive auditing protocol is necessary to maintain bidder privacy. The protocol is very beneficial to the system, though it certainly adds a non-negligible amount of complexity and overhead. In comparison, AUTHAUCTION sacrifices bidder anonymity to allow each and every member to independently audit the entire auction procedure. Again, this underlines the importance of deniability to VEX versus the importance of decentralization to AUTHAUCTION.

An essential difference between VEX and AUTHAUCTION is the assumption that collusion will not occur. Suppose the seller colluded with the auctioneer at the beginning of the *decommitment round*. For each bidder, the seller could construct a bundle of commitment data containing all fake information besides that one bidder's actual commitment datum. The bidder will see the signed bundle, find his own information untampered, and proceed to submit the corresponding decommitment datum. Of course, the seller cannot know what the bid was before the decommitment datum is submitted, but he can nonetheless construct a bundle of bids that are reasonably low to ensure that the targeted bidder is likely to "win" the auction. In such a way, the seller can make every bidder believe that they have won. The bidders are likely to discover this when all but one realize that they were never given the ad space sold in the auction, however there does not appear to be another way this can be detected.

One less rigorous feature of AUTHAUCTION is that it frustrates attacks when the Server ignores or drops a member's bid. AUTHAUCTION requires that every user know of every other user. This makes it simple for any member to determine whose bid is missing when such a situation arises. The same cannot be said for VEX whereby not only can no member determine whose bid is missing, but it is possible that no member would realize that a bid was missing in the first place.

7 Criticalities

7.1 Subtleties

A few essential subtleties need be pointed out to ensure that simple but catastrophic errors are avoided by anyone implementing this cryptosystem.

The first subtlety is the necessity that every member of an Estate participates in every auction. If a member does not wish to bid on an item, the same bidding protocol is to be carried out, the only difference being that $b = 0$. A user not responding is indistinguishable from the Server ignoring the user, thus we necessitate that every member submit a datum.

The onus of reporting any error whatsoever is completely on each and every member of an Estate. If any error is encountered, the user is to report the error to every other member of the Estate through out-of-band communication. This is the only way that every member can be sure that an auction completed without errors.

To keep track of how much a user has bid, an implementation may opt to give each user a finite number of coins and have only the winner of an auction have the appropriate number of coins withdrawn from their sum. To avoid trusting the Server, all coin-amount calculation ought to be done by each client individually. The number of coins should only be decremented if the user can verify for himself that the winner of the auction truly was the winner. Thus, the number of coins each user has should not be stored on the Server so as to prevent the users from naïvely trusting the Server's calculations.

7.2 Vulnerabilities

It is just as important to indicate what guarantees a cryptosystem does not make as it is to indicate those that it does make. Certain design choices have led AuthAuction to be unable to make certain guarantees to the user. These missing guarantees that are security related are listed as vulnerabilities.

One essential part of every auction process is the waiting period between Phase #1 and Phase #2 whereby every member is given time to copy every other user's bid commitment into the client software. If this waiting period is skipped, users cannot download the commitment data and thereby lose the ability to verify other member's bids in Phase #2. If this situation occurs, it is required that the affected user(s) make this known to the rest of the members.

The process of cheating mitigation contains a vulnerability within itself. Any user has the ability to claim that they have been misinformed by the Server, a verification procedure failed, etc. No other user has any ability to prove whether the whistleblower is telling the truth or not. Thus, the only procedure regardless of the true intentions of the whistleblower is to repeat the process until it completes with no issues reported by any user. Thus any single user can indefinitely stymie the procession of their Estate.

8 Future Work

It is possible that this system can be orchestrated through a public means whereby identity information and communication are all provided by some trusted source. We highlight that trust should not by any means be taken lightly. One practicable candidate for this method of auction conduction is Twitter. It may be feasible for users to simple trade Twitter handles at the creation of an Estate, thereupon conducting all in-band communication through Twitter messages. One notable obstacle that stands is that *tweets* (Twitter messages) are limited to 140 Unicode characters. A potential solution would employ the use of message splitting in an unambiguous manner.