IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Preserving Privacy and Improving Application Security of an e-Auction system using Intel SGX

[1]Anil Kumar Tegala, [2]Nani Saragada

[1]Student (M. Tech) Cyber Security & Data Analytics, [2]Student (M. Tech) Cyber Security & Data Analytics
[1]Dept. of Information Technology & Computer Applications,
Andhra University College of Engineering, Visakhapatnam, India

Abstract**:** With the growth of Internet usage, e-Auction systems have become popular and became an electronic market place for many. They enable users to sell and purchase goods on a large scale. There are several challenges in implementing e-Auction systems. Stakeholders located at geographically different locations can cheat auction systems. Furthermore, fake auction systems might target people and lure them. Though there are a few secure e-Auction systems in use that focus on preventing fraud from sellers and bidders, challenges still exist. Fraudulent auction transactions happen when the auctioneer is compromised.

This paper proposes a fraud-resistant auctioneer model, which provides privacy and security using Intel SGX in a Trusted Execution Environment (TEE). It has less attack surface and provides better attack protection to the application. This paper discusses the design and implementation of the proposed secure and trustable e-Auction system. The proposed model, in a typical attack scenario: i) can verify the authenticity of the data and security using keys and digital signatures, ii) protects from exposing the sensitive information of buyers and sellers and, hence privacy is protected, iii) the winner decision making process is made by TEE, so is trustable.

Index Terms**- e-Auction, Intel SGX, security, privacy, enclave, TEE.**

I. INTRODUCTION

Online auction (e-Auction) has become popular for conducting auctions for retailing and purchasing products through the web. An e-Auction is accessible to everyone. Online auctions have many advantages over traditional auction systems such as geographic proximity, logistical limitations, physical space, and target audience. An e-Auction can be conducted on a large scale as the internet accommodates a large network of users to participate.

As shown in fig. 1, there are two categories of auctions: open and sealed auctions. The most popular auction is English Auction, also known as Forward Auction. Goods and services are offered for bidding in an open ascending manner where a bidder places bids at an increasing price. The bidder who places the highest bid is the winner.

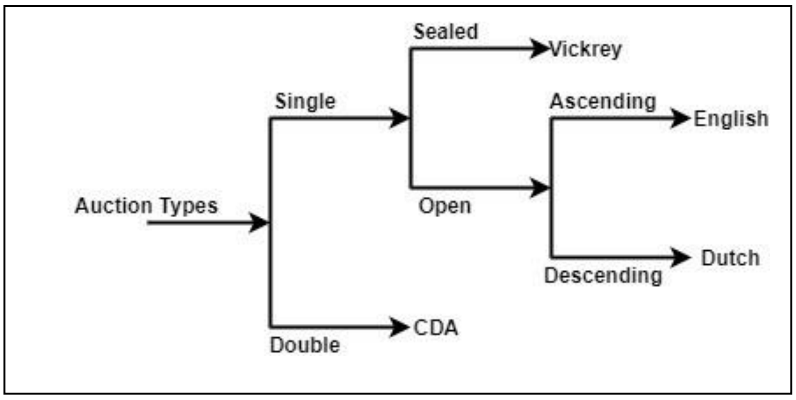


Figure 1: Auction Types

Another auction which is similar to Forward auction and it is quite reverse to it, is called Dutch Auction. It is also known as Reverse Auction. In this open auction, bids will be placed in a descending manner, and the price will be decreased until the bidder intends to pay the price he wanted to. Sealed Bid auction is another type of auction in which auctioneer opens all the bids and sells the product for the highest bid (price) placed.

In Vickrey Auction, which is also known as the second- price sealed-bid auction, the highest bidder and winner will pay the bid in which the second-highest bidder had placed. Continuous Double Action (CDA) is a special type of auction which benefits many sellers and buyers. Sellers and Buyers will place the bids continuously for the sale and purchase of a product. CDA scenario can be seen in Stock Markets.

II. SECURITY, PRIVACY AND TRUST IN E-AUCTION

A. Frauds in online auction

Even though e-Auction systems have been designed to ensure security, frauds still exist while conducting auctions online. Some of the known frauds in the online auction are

- *Shielding:* An artificial highest bid will be placed just before the bid close time, which discourages the other participants (bidders) in the auction [15].
- *Shilling:* A bidder will try to inflate the price of the selling bid. It is a technique that benefits the seller to sell the product for a more profitable price [16].
- *Sniping:* Some bidders will be refrained from making bids until the close time of bidding arrives and makes a bid in such a manner that prevents other bidders from bidding at the closing time [16].
- *Siphoning:* An outsider who is not a participant in auction observes an auction and makes an offer to the bidder directly for a similar product to the current bidding [16].

B. Security features in e-Auction system

Some of the security features an e-Auction system is expected to have are as follows*:*

- *Nonrepudiation:* Once a bid is placed, the bidder must not be able to repudiate it.
- *Un-forgeable bids:* Other bidders should not forge bids impersonating genuine bidders.
- *Publicly verifiable:* Some publicly available information such as bidder registration, bidding and winner / loser can be verified by all parties participating in e-auction.

- *Robustness:* Bidders must follow the auction protocol and auction process will not be affected.
- *Efficiency:* Registration, signing a bid, verifying and winner determination will be effectively handled.

C. Trust in online auctions

An e-Auction model must provide trust to the bidders and protect their data from corrupt or malicious auctioneers. Most of the auction models isolate auctioneer from bidding, and auctioneer only conducts auctions online. They do not have any privileges to change auction protocol in the securely designed e-Auction model. In many cases, trusted third parties are used to establish trust between auctioneer and seller/buyer.

III. INTEL SGX

Intel designed a hardware-assisted Trusted Execution Environment (TEE) to help minimize the attack surface with Intel Software Guard Extensions (SGX) [17]. It is a set of extensions to Intel architecture, which provides confidentiality and integrity to secure sensitive computation performed on a computer where all the privileged software (like Kernel, Hypervisor) is potentially malicious.

Intel SGX protects an application's secrets from malicious software by putting code in an isolated memory region (Protected Memory) called Enclaves. The protected memory, which is about 32MB to 128MB, is reserved from systems physical RAM and then encrypted. In Intel SGX, secrets remain secrets even if the application, OS, VMM, and BIOS are compromised.

Many solutions benefit from the additional protection provided by Intel SGX. Solution examples include AI and ML processing, key management, proprietary algorithms, and protection of biometrics.

A. Developing Intel SGX Application

An Intel SGX [19] application can be built with two parts: an untrusted component and a trusted component. The trusted part contains Enclave – which contains private data of computation and the code that operates on it. The Untrusted component is the rest of the application.

Any application or memory region, which was not protected, will be considered as untrusted by the Intel SGX [17]. A total of 1-n enclaves can be created that can support distributed architectures.

When an application needs to work with the secret information in Enclave, it creates an enclave, which is placed inside protected memory. It then calls a trusted function to enter the Enclave, where the secrets can be viewed. The processor denies all other attempts to access information in enclave from outside the enclave, even those made by privileged users. This prevents secrets in the enclave from being exposed.

B. Intel SGX Security Model

The Intel SGX [18] instructions will build and execute the enclave into a protected encrypted memory region with restricted entry/exit points defined by the developer. This helps prevent data leakage. Enclave code and data inside the CPU boundaries, and enclave data written to memory is encrypted and integrity checked, thus providing some assurance that no unauthorized access or memory snooping of the enclave occurs.

C. Some attack surface for Intel SGX

Intel SGX can reduce the attack surface of an application. Fig. 2 shows the difference between surfaces with and without using Intel SGX Enclaves.

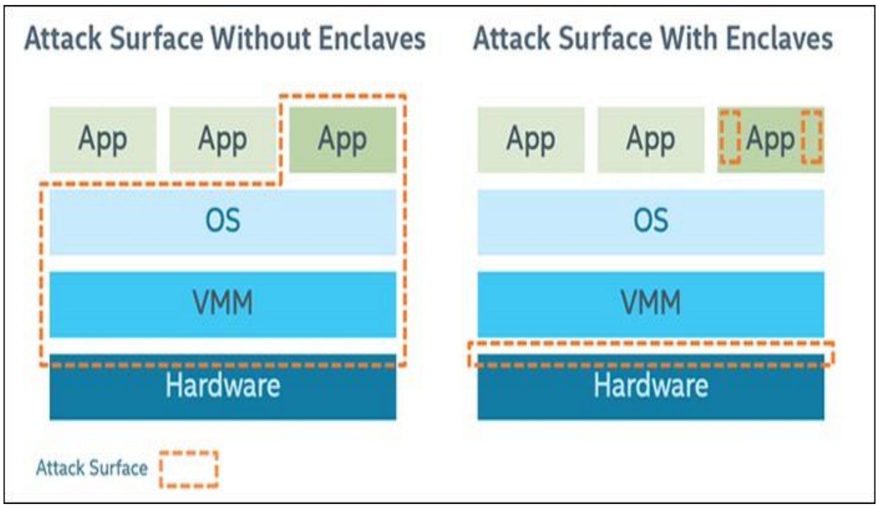


Figure 2: Software Guard Extension attack surface areas [21]

Intel SGX helps in protecting applications from several known attacks, both hardware, and software.

- The advantage is that the Enclave memory cannot be read from outside the enclave irrespective of the privileges and the CPU Mode.
- Another advantage is that the Enclave cannot be entered using Function calls, jump statements, manipulation of registers, and manipulating stacks. The only way to call the enclave function is through a new instruction after performing several protection checks.
- Data Isolation within enclaves can only be accessed by code that is shared by enclave.

D. Data Sealing

Intel SGX provides a capability called Data Sealing. When data is sealed, it is encrypted in the enclave, using an encryption key that is derived from CPU. This encrypted data block, also called the sealed data, can only be decrypted or unsealed, on the same system where it was created. The encryption itself provides assurance of confidentiality, integrity, and authenticity of the data.

IV. EXISTING ONLINE AUCTION MODELS AND PROBLEM MOTIVATION

A. Existing e-Auction Models

Auction models infrastructure allows the set-up of distributed online auctions with flexible, fraud-resistant mechanisms. Reliable process directions are developed in contrast to cryptographic or hardware approaches. This system is composed of functional modules, it provides evidence of activities done by bidders and auctioneers, making fraud detection possible [11].
A real-time auction system was designed for Tea Auction, where Bidders can set up alarms in this system when it is close to the maximum price to get good results. This reduces more time for bidders can be utilized for auctioning other items [12].
A theoretical model, e-Auction, was designed for two auction types: forward and reverse auctions. The use of trust solved a dependency issue that arises during the e-auction transactions within the registration phase. Each seller (or) buyer should register at the auction site by using an Id, which is used for the authentication to access the auction and allowed access for performing operations. By using a high level of trust and security, they are protecting e-Auction from malicious sellers and buyers [13].
uAuction is an online auction system, and it presents the analysis, design of the auction system by UML diagrams. It is an open-source approach; researchers can expand upon this system, and it is grounded in object-oriented techniques. uAuction also provides the ability to conduct various types of tests [14].

B. Problem Motivation

Many applications and technologies have started using the emerging Software Guard Extension. Companies and services are porting their applications and services to this Trusted Execution Environment (TEE) [1]. Even secure apps like SecureKeeper [7], Ryoan [6], Scone [5] have become more secure after embedding their applications with Intel SGX. E.g. Tor - A Browser, which provides security and anonymity for users, has become more secure after it is blended with Intel SGX as SGX-TOR [8]. Client-side JavaScript has no trust as it is ubiquitous. TrustJs [9] – A JavaScript framework that is integrated with Intel SGX provides trusted execution of JavaScript at the client-side.
Intel SGX also can be attacked with several attacks like side-channel attacks [3], cache attacks [4], but these are quite minimal in nature.
Looking at several secured and cryptographic e-Auction models available and the challenges in security, privacy, and trust faced, this paper focuses on auctioneer functionality in the e-Auction system.

V. PRESERVING PRIVACY IN E-AUCTION USING SGX

A. Application Design

This section discusses the architecture of the e-Auction application while using Intel SGX environment. The e- Auction model discussed in this paper has the following functionalities.

- Identify sensitive information in e-Auction model such as bid values along with the respective bid id's placed by bidders.
- Identify which code belongs to the Enclave.
- Identify the Enclave boundary.
- Identify the interaction of the trusted and untrusted parts with each other.

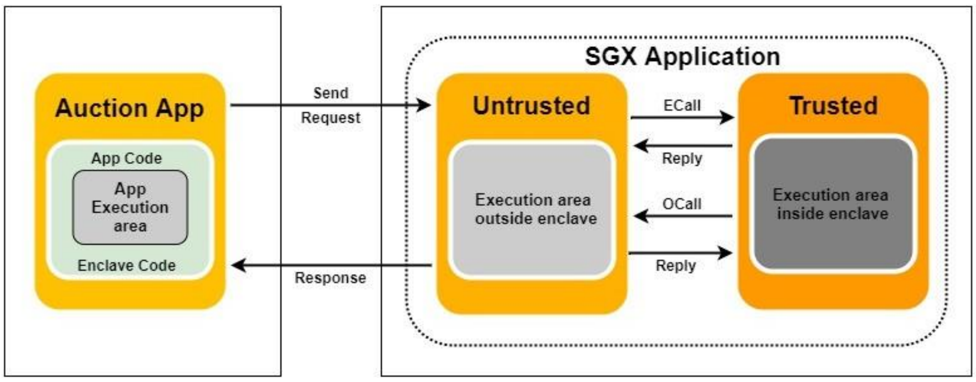


Figure 3: Application flow control

The Auction App and Enclave application consist of three components as shown in fig. 3.

a. Auction App is a client application which is executed in an Intel SGX Machine.
b. Application's Untrusted Part
c. Application's Trusted part

Fig. 4 shows the following communication flow between the auction model and the trusted and untrusted parts.

i. Auction App sends the request to the Untrusted area.
ii. The Untrusted part sends an ECALL request to the trusted part.
iii. The Trusted part executes and sends the reply to the untrusted component.
iv. The Untrusted part sends the response (OCALL) to the Auction App.

B. Implementation and Methodology

The proposed model has Auction App (which acts as an auctioneer) and conducts forward and reverse auctions and declares winner by generating winner using secured Enclaves. This section explains the steps involved.

1) Module 1: User Registration and Auction Participation

Users/bidders get registered to participate in any auction conducted by the auctioneer. Users, who do not have registered id, cannot participate in bidding. Before entering into the auction, Selling and Bidding users who have allotted 'Id' place their encrypted bids (bid_value) using verifiable signature (bidder_sign).

Different participants in the Auction App are:

- *Seller:* A seller offers a list of items (Product) for sale.
- *Bidder (Buyer):* A bidder submits a bid for an item listed by the seller.
- *Auctioneer (App itself):* Hosts create the auction and are responsible for conducting auction proceedings according to the auction rules. (App implements all auction protocols)

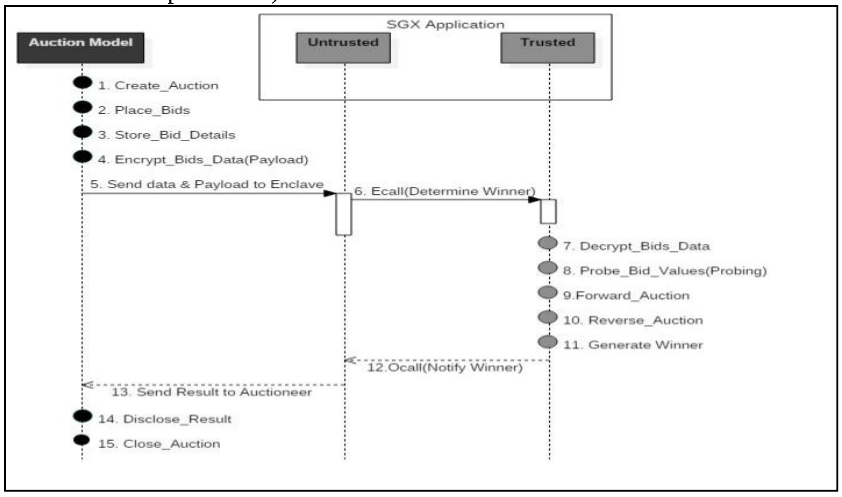


Figure 4: Data communication between Auction App and Enclave

2) Module 2: Data communication between Auction App and Enclave

In this module, the Auction App (client) sends the payload and data to the Enclave (SGX) using a secure design it has. Data can be shared between Enclave and Auction App using an enclave id (eid). The enclave id is the randomly generated symmetric key. A new enclave id will be generated for each auction, which is conducted through the Auction App. After receiving payload, the enclave will be executed through ECALL to determine auction (Forward or Reverse) winner in a secure way. The winner is notified using OCALL from Enclave to Auctionapp.

C. Experimentation and Analysis

The model is tested on Dell OptiPlex 5040 - MT - Core i7 6700 3.4 GHz - 8 GB CPU, which supports Intel SGX and executed using Microsoft Visual Studio Professional 2015. We created a console auction application using visual C++ and integrated that app with Intel SGX Application [20].
The flow of application is shown in the fig. 5. The following are the interacting code files for e-auction in SGX:

i. Auctionapp.cpp – contains Auction Code
ii. Enclave.cpp – contains ECALLS & OCALLS
iii. Enclave.edl – ECALLS and OCALLS will be defined in this enclave definition language file.

On running Auctionapp.cpp (Auctioneer), it asks you to decide whether you want to participate in a Forward_Auction() or Reverse_Auction(). Based on the selection, Users(bidders) can place their bids along with their id's for the product proposed for sale. After Placing bids through Auction.cpp, it sends Payload and data to Enclave using an ECALL encrypt_bidvalues(). The Payload and auction data are encrypted and sent to Enclave to protect Sensitive information. This is done as its contents can be seen after disassembling it if not encrypted.

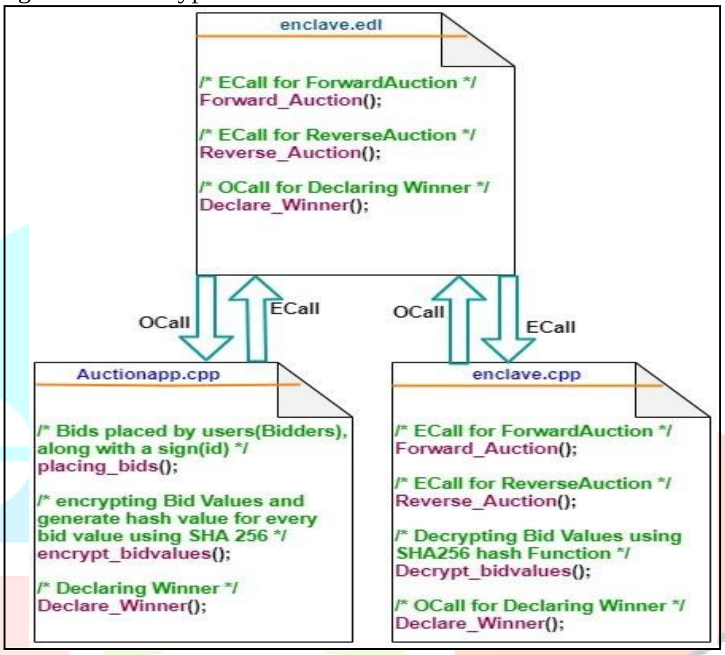
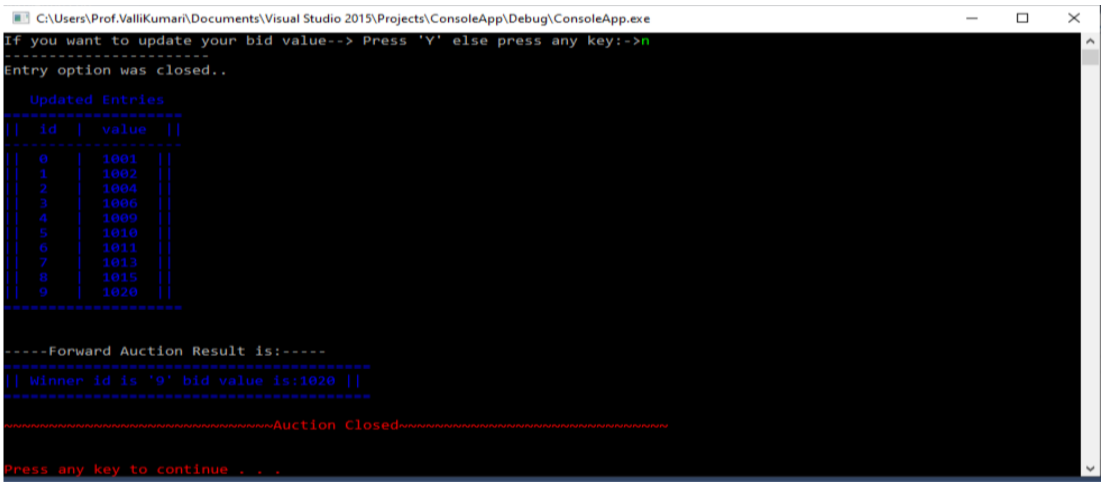

enclave.edl
/* ECall for ForwardAuction */
Forward_Auction();
/* ECall for ReverseAuction */
Reverse_Auction();
/* OCall for Declaring Winner */
Declare_Winner();
OCall
ECall
OCall
ECall
Auctionapp.cpp
/* Bids placed by users(Bidders), along with a sign(id) */
placing_bids();
/* encrypting Bid Values and generate hash value for every bid value using SHA 256 */
encrypt_bidvalues();
/* Declaring Winner */
Declare_Winner();
enclave.cpp
/* ECall for ForwardAuction */
Forward_Auction();
/* ECall for ReverseAuction */
Reverse_Auction();
/* Decrypting Bid Values using SHA256 hash Function */
Decrypt_bidvalues();
/* OCall for Declaring Winner */
Declare_Winner();

Figure 5: Execution flow between Auction App and Enclave

Once Payload and data are sent into Enclaves, An ECALL Decrypt_bidvalues() decrypts and probes bid data and values to check if any tampering or fraud has happened or not. For this behavior detection, cryptographic protocols are programmed. If any manipulation is done with auction data, it is ignored and is not considered for winner determination. Otherwise, the code in enclave will make an ECALL Declare_winner() as in figure 6, which analyses the data and determines the winner.
Once the Winner is declared inside Enclave, it notifies the winner by making an OCALL Notify_winner() to Auction App. The Auction App then discloses the result and closes the auction.



```
C:\Users\Prof.ValliKumari\Documents\Visual Studio 2015\Projects\ConsoleApp\Debug\ConsoleApp.exe
If you want to update your bid value--> Press 'Y' else press any key:->n
-----------------------
Entry option was closed..

   Updated Entries
====================
||  id  |  value  ||
--------------------
||  0   |  1001   ||
||  1   |  1002   ||
||  2   |  1004   ||
||  3   |  1006   ||
||  4   |  1009   ||
||  5   |  1010   ||
||  6   |  1011   ||
||  7   |  1013   ||
||  8   |  1015   ||
||  9   |  1020   ||
====================


-----Forward Auction Result is:-----
==========================================
|| Winner id is '9' bid value is:1020 ||
==========================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Auction Closed~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


Press any key to continue . . .
```

Figure 6: Declaration of Winner

However, the sensitive information of the e-Auction app placed inside is not accessible to other legitimate users, including auctioneer. However, disassembling the enclave can expose the secrets. For that reason, SGX [22] provides a capability called data sealing. The sensitive information we put inside the enclave must be sealed to enable protection. The sealed data will be written to untrusted memory or can be stored in any media to access it. Otherwise, SGX removes all the enclaves and its contents once the program exits or the system is shut down.

VI. CONCLUSION AND FUTURE WORK

We have leveraged Intel SGX for the e-Auction model to provide better security and preserve sensitive application information. The proposed solution improves security, privacy, and trust concerns than other e-Auction models pose. This design can be used for other existing e-Auction models, including the CDA models.

REFERENCES

[1] Ofir Weisse, Valeria Bertacco, Todd Austin, Regaining Lost Cycles with Hot Calls: A Fast Interface for SGX Secure Enclaves, The 44th Annual International Symposium on Computer Architecture, 81-93, 2017.

[2] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, Carlos Rozas, *Intel® Software Guard Extensions (Intel®SGX) Support for Dynamic Memory Management Inside an Enclave,1-2, HASP 2016.*

[3] Oleksii Oleksenko, Bohdan Trach, Robert Krahn, Mark Silberstein, Christof Fetzer, *Varys: Protecting SGX enclaves from practical side- channel attacks*, {USENIX} Annual Technical Conference, 227-240, 2018.

[4] Ferdinand Brasser, Urs Muller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, Ahmad-Reza Sadeghi, *Software Grand Exposure: SGX Cache Attacks are Practical*, 11th USENIX Conference on Offensive Technologies, 1-12, 2017.

[5] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, and Mark L Stillwell, David Goltzsche, Dave Eyers, Rüdiger Kapitza, Peter Pietzuch, Christof Fetzer, *SCONE: Secure Linux Containers with Intel SGX*, 12th {USENIX} Symposium on Operating Systems Design and Implementation, 689-703, 2016.

[6] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, Emmett Witchel, *Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data*, 12th {USENIX} Symposium on Operating Systems Design and Implementation, 533-549, 2016.

[7] Stefan Brenner, Colin Wulf, David Goltzsche, Nico Weichbrodt, Matthias Lorenz, Christof Fetzer, Peter Pietzuch, Rüdiger Kapitza, *SecureKeeper: Confidential ZooKeeper using Intel SGX*, 17th International Middleware Conference, 1-13, 2016.

[8] Seongmin Kim, Juhyeng Han, Jaehyung Ha, Taesoo Kim, Dongsu Han,
Enhancing Security and Privacy of Tor's Ecosystem by using Trusted Execution Environments, 14th {USENIX} Symposium on Networked Systems Design and Implementation, 145-161, 2017.

[9] David Goltzsche, Colin Wulf, Divya Muthukumaran, Konrad Rieck,Peter Pietzuch, Rüdiger Kapitza, *TrustJS: Trusted Client-side Execution of JavaScript*, 12th EuroSys Conference, 1-6, 2017.

[10] Swarup Chandra, Vishal Karande, Zhiqiang Lin, Latifur Khan, Murat Kantarcioglu, Bhavani Thuraisingham, *Securing Data Analytics on SGX with Randomization*, 22nd European Symposium on Research in Computer Security, 352-369, 2017.

[11] Júlio Da Silva Dias, Carlos Roberto De Rolt, Thiago Souza Araujo, *Secure Distributed Auction Infrastructure*, IADIS International Conference, 125-132, 2007.

[12] P. Hemantha Kumar, Gautam Barua, *Design of a Real-Time Auction System*, 4th International Conference on Electronic Commerce Research, 1-9, 2001.

[13] Hameed Ullah Khan, Abdullah M. Al-Faifi, Diab Mahmoud Diab, *Theoritical Model for eAuction*, IJCSNS International Journal of Computer Science and Network Security, 116-120, 2012.

[14] N. Majadi, J. Trevathan N. Bergmann, *uAuction: Analysis, Design, and Implementation of a Secure Online Auction System*, 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 278-285, 2016.

[15] Kazi Mamun, Samira Sadaoui, *Combating Shill Bidding in Online Auctions, International Conference on Information Society, 170-176, 2013.*

[16] Fei Dong, Sol Shatz, Haiping Xu, *Combating online in-auction fraud: Clues, Techniques and Challenges*, Computer Science Review, 245- 258,2009.

[17] Intel SGX Explained https://eprint.iacr.org/2016/086.pdf

[18] Intel Software Guard Extensions Programming Reference https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf

[19] Intel Software Guard Extensions (Intel SGX) https://software.intel.com/en-us/sgx

[20] Intel Software Guard Extensions (Intel SGX) SDK https://software.intel.com/en-us/sgx/sdk

[21] Intel SGX Tutorial Series https://software.intel.com/en- us/articles/intel-software-guard-extensions-tutorial-part-1-foundation

[22] Intel SGX for Windows https://software.intel.com/en- us/articles/getting-started-with-sgx-sdk-for-windows