Secure Brokered Delegation through DelegaTEE

Moritz Schneider*, Sinisa Matetic*, Ari Juels†, Andrew Miller‡ and Srdjan Capkun*
* ETH Zurich, {moritz.schneider, sinisa.matetic, srdjan.capkun}@inf.ethz.ch,
† Cornell Tech, juels@cornell.edu,
‡ UIUC, soc1024@illinois.edu

***Abstract*—We introduce DELEGATEE, a system that enables users to delegate their rights and resources in existing services safely and selectively to others—without ever revealing their access credentials to third parties. By using hardware-enforced Trusted Execution Environments, DELEGATEE enables contextually rich delegation policies that were previously unachievable.**

I. INTRODUCTION

Delegation, the ability to share capabilities or privileges selectively to other entities, is a well-studied concept in access control. Delegation remains mostly unsupported in today's online services, however. Most web-based financial services lack support for any kind of delegation, while other services, such as Facebook, support it in a limited and coarse-grained way. Facebook allows a user to delegate to a third-party application the authority to post to the user's wall, but not to impose a limit of, say, three posts per day.

The ability to delegate access to existing online accounts and services, *safely and selectively*, could give rise to new forms of cooperation among users. One example is flexible sharing (or resale) of digital content, such as streaming services like Netflix. Another is the outsourcing of online tasks, such as replying to email, to remote workers. Yet another is delegation of access to financial service accounts, such as Paypal. Given such capabilities, ordinary users could play a role in broadening global financial inclusion.

One obstacle to this vision is that service providers today exert almost complete control over resource sharing by their users. If users want to share data or delegate access to services in ways not natively supported by their service providers, they must resort to sharing credentials directly. This is a dangerous practice: an abused shared credit-card number can mean significant monetary loss, while an abused shared password can result in high charges, service termination, and even legal jeopardy. Such dangers naturally deter against many forms of online content and service sharing.

The goal of our work is to enable delegation by an Owner, i.e., the entity that has access to a resource or a service, to a Delegatee, i.e., a borrower of that resource or service, while achieving several key properties. First, we want delegation to be *fine-grained*, i.e., to limit the capabilities of the Delegatee in carefully specified ways. We also want it to be *trust-limited*: the Owner need not trust the Delegatee with direct access to the delegated credential, the Delegatee need not trust the Owner to make the credential available, and neither need to trust any third party with its credentials. Finally, we want delegation to be *provider-independent*: there should be no need for explicit support (or even awareness) by service providers. This puts Owners in full control of delegation of their resources and services.

We refer to the new kind of flexible and powerful sharing of resources embodied in these properties as *brokered delegation*. Brokered delegation has only recently become broadly practical thanks to recent advances in *trusted execution environments* (TEEs), a hardware-based application-security technology that is available today in Intel chipsets under the name Software Guard eXtensions (SGX) [6] as well as the Keystone project in RISC-V based systems [11] and ARM TrustZone [2].

To demonstrate the potential of brokered delegation, we present DELEGATEE. DELEGATEE provides brokered delegation for many existing web services with rich, contextually-dependent access-control policies. Its use of TEEs enables it to do so while concealing and thus protecting Owner credentials. We develop several application prototypes to demonstrate how the brokered delegation in DELEGATEE can potentially give rise to new markets: secure outsourcing of personal and commercial microtasks, tokenization (i.e., creation of fungible, tradeable units of resource), resale of resources and services, and new payment methods - all without changes to legacy infrastructure. One of the key features of DELEGATEE is its high degree of provider-independence: it *requires no changes whatever to the service managing the resource or to users' accounts*.

DELEGATEE also demonstrates a broader insight into the security consequences of trusted hardware: TEEs can fundamentally subvert access-control policy enforcement in existing online services. Depending on the application, DELEGATEE can either enrich a target service or undermine its security policies (or both). For example, reselling a paid subscription service in regions where the service is intentionally restricted undermines the service's security policy, while delegating access to office tools such as mail and calendar to administrative assistants can enrich the capabilities and usability of the service itself. Brokered delegation can also facilitate violations of web services' terms of use. Users may thereby circumvent *mandatory access control* (MAC) policies, reducing them to *discretionary access control* (DAC). DELEGATEE subverts full mediation. We refer the interested readers to the full paper for a more in-depth view on the topic [13].

By enabling new forms of sharing and cooperation among users, DELEGATEE evokes the technology-fueled resource-sharing models for physical resources pioneered by Airbnb and Uber. These companies have challenged legal and regulatory frameworks while creating and delivering appealing new services. We view DELEGATEE as a catalyst for a new class of such contributions to the sharing economy.

II. Motivation and Problem Statement

A. Motivation

There are two major motivations for our work: To demonstrate the many settings in which brokered delegation gives rise to new functionality, and to demonstrate how (for good or bad) TEEs can transform practically any mandatory access control policy in an online service into a discretionary one. Our four different application scenarios illustrate both motivations, without requiring the explicit support (or even knowledge) of the service providers. We stress that our motivation lies in extending legacy and current systems with delegation capabilities without security implications for the involved parties. Many services could implement native and extensive delegation policies themselves, however, this is often not of general interest to them and not in line with their business strategies.

Mail/Office. Full or restricted delegation of a personal mailbox or other office tasks can be appealing for many reasons. These include a desire to delegate work to administrative assistants (e.g., read-only access, send mail only to a specific domain) or to allow limited access to law-enforcement authorities (e.g., read emails from a certain time window relevant to a court case). The first is especially valuable for virtual-assistant services, which outsource office tasks off-site. Existing services offer some delegation capabilities, such as calendar access by a third person, however these may be limited and usually do not cover the delegation on the full scope of the service offering. Today, these services require users to completely share their credentials, a dangerous practice that discourages many potential users.

Payments. Virtually all payments, cash and cryptocurrencies excepted, happen through intermediaries. Users may naturally desire a richer array of choices of these intermediaries. Consider, for example, a payment system where the users pay using each others' bank accounts, credit cards, or third-party providers (e.g., PayPal). This can have large benefits regarding cost-saving, business operations, and anonymity guarantees.

Imagine that a company wants to allow its employees to execute online purchases with the company credit card or PayPal, but restricted to a certain limit per expenditure and specific merchants. Currently, this cannot be done since access to the card details or PayPal credentials allows users to execute arbitrary payments. Companies therefore typically provide such information only to a few employees who then execute payments for the rest, resulting in a highly inefficient process. Corporate credit cards exist and support more extensive reporting and control over the spending, e.g. tied to specific employees or expenditures, however, rich and contextual access control policies are still lacking and would be difficult to implement.

Delegation of payment credentials can also enable direct cost-savings for the end user. An example online system based on this premise is Sofort. Sofort works as an internet payment middleman, with lower transaction fees than for credit cards. Sofort pays merchants for clients' online purchases and is repaid by clients via bank transfer. To guarantee repayment, Sofort requires users to share their e-banking credentials with the service, a practice that clearly raises security and privacy risks.

Full Website Access. The most versatile form of delegation is delegation for arbitrary existing web services, which typically authenticate user accounts through password challenges and then cookies over HTTPS. This model includes access to users' social networking sites, video services, online media such as news and music, and general website content available only to registered users. One appealing example from the academic world is *Sci-Hub*. The site bypasses publishers' paywalls using a collection of credentials (user IDs and passwords) belonging to educational institutions which have purchased access to the journals." Many anonymous academics from around the world donate their credentials voluntarily [3]. Some services, such as Netflix and various news sites, already offer users the ability to log in from different devices. Users can thus share their subscriptions by sharing credentials, but only in a dangerous all-or-nothing manner. More fine-grained, e.g., service-specific, and secure delegation could facilitate much broader sharing (for good and bad).

Sharing Economy. The examples above involve an Owner delegating credentials to known Delegatees, e.g., friends or colleagues. However, Owners can also offer access to their services on an open market to a wide range of potentially pseudonymous or anonymous Delegatees. This would result in a shared economy in which Owners sell time-limited and restricted access to their accounts in return for other services or financial compensation. For example, users could sell access to Netflix accounts on an open market. They could also sell space in their social networking accounts to advertisers; e.g., a user could sell the ability to post in her name, enabling an advertiser to target her social network. The right to post could be restricted to a particular volume and type of content to prevent abuse by advertisers.

B. Problem Statement

If service providers regularly offered richly featured native delegation options, there would be no need for **brokered delegation**. Most do not, however, usually for business or regulatory reasons. Our work aims to change this situation fundamentally — DelegaTEE empowers users to delegate their authority, making use of any existing internet service, such that:

- The Owner's credentials remain confidential.
- The Owner can restrict access to her account, e.g., regarding time, duration of access, no. of reads/writes, etc.
- The system logs the actions of Owners and Delegatees so that post-hoc attribution of their behaviors is possible (as a means of resolving disputes).
- The system minimizes the ability of a service to distinguish between access by the Delegatee and that of the legitimate Owner, thus, preventing delegation.

III. DelegaTEE

The main idea behind the DelegaTEE system is to send the Owner's credentials (such as usernames and passwords) to a Trusted Execution Environment (TEE) that implements the delegation policy. The Delegatee communicates with the resource (web service) indirectly, using the TEE as a proxy. In this section, we briefly introduce background on TEEs, then present the DELEGATEE system design.

A. TEEs and Intel SGX Background

Modern TEE environments, most notably ARM TrustZone [2], Intel SGX [6], RISC-V's Keystone [11] and Sanctum [7], enable isolated code execution within a user's system. Intel introduced SGX in the 6th generation of its CPUs as an instruction set architecture extension. Like TrustZone, an older TEE that permits execution of code in a "secure world" and is used widely in mobile devices, SGX allows isolated execution of the code in what is referred to as secure *enclaves*. The SGX security architecture guarantees enclave *isolation*, using protective mechanisms enforced in the processor, from all software running outside of the enclave. The control-flow integrity of the enclave is preserved and the state is not observable. The code and data of an enclave are stored in a protected memory area that resides in Processor Reserved Memory (PRM) [14].

In TrustZone, the transition to the secure world involves a complete context switch. In contrast, the SGX's secure enclaves only have user-level privileges and a special instruction is invoked to switch between the enclaves and the OS.

Additionally, SGX includes a key feature unavailable in TrustZone, called *remote attestation*. Attestation is the process of verifying that enclave code has been properly initialized. In *remote attestation* the verifier may reside on another platform. A special system service called Quoting Enclave signs the attestation statement, which includes the enclave measurement (code and date), with the secret key derived from the embedded platform key. The verifier checks the signature with the help of an online attestation service run by Intel, and if successful, can be sure that a specific piece of code is running in a genuine SGX environment.

In summary, the main protective mechanisms supported by SGX are: runtime isolation [14], sealing [1], and attestation [9], [6]. The SGX architecture enables the app developer to create multiple enclaves for security-critical code, protecting it from malicious applications [16], a compromised OS, virtual machine manager [5], or BIOS [10], and even insecure hardware [8] on the same system. We relegate further details below; for an in-depth treatment of SGX, see [6]. Note that recent research has shown that SGX is susceptible to various side-channel [4] as well as speculative execution attacks [15], however, we consider these out of scope in this work.

Recently, academia has proposed two new TEE platforms based on the open-source RISC-V instruction set architecture: Keystone [11] and Sanctum [7]. They are both based on a trusted Security Monitor that manages isolation and attestation of enclaves, and they support similar guarantees as Intel SGX.

B. System Design

Three distinct parties participate in DELEGATEE: credential *Owner(s)* A, *Delegatee(s)* B, and *service(s)* G. Additionally, the system distinguishes 2 data types: *credential(s)* C and *access control policy(ies)* P. Owners and Delegatees are generically referred to as *users*.

The system supports a potentially large population of credential Owners $A_1...A_n$ (henceforth referred to as Owners) and Delegatees $B_1...B_n$. In general, the Owner A_i has access to a service G_k. The Delegatee B_j does not have access to the service, but she can get access by using credentials C_x of the Owner A_i. However, the Owner A_i does not want to reveal the credentials to the Delegatee B_j. The Owner A_i wants her credentials to remain confidential and used only by an authorized Delegatee. Additionally, the Owner wants to restrict access to the services that she enjoys (i.e., G_k) according to an access control policy P_{ijxk} specific to this delegation relationship. P_{ijxk} defines an policy involving Owner $A_\mathbf{i}$, Delegatee $B_\mathbf{j}$, credentials $C_\mathbf{x}$, and service $G_\mathbf{k}$. The type and structure of the access control policy depend on the service that the Owner delegates.

In our system, a Delegatee directly coordinates with the Owner to gain access to a specific service. The Intel SGX enclave is hosted by a third party, called the Central Broker. The Central Broker is responsible to host the enclave but he is not implicitly trusted, e.g., the OS on the broker can be compromised or a malicious administrator is present. More in-depth discussion follows in Section IV. The steps to execute secure credential delegation, also given in Figure 1, are:

1) Both the Owners ($A_1...A_n$) and the Delegatees ($B_1...B_n$) need to register with the system to acquire unique login information (username and password). After registration, both Owners and Delegatees can execute credential delegation for service access.
2) The Owners $A_1...A_n$ now establish a secure channel to the system (using the ordinary web PKI) and start storing the credentials $C_1...C_n$ for specific services $G_1...G_n$. The variety of credentials that can be stored depends on the supported services.
3) The Owners $A_1...A_n$ may agree directly with the Delegatees $B_1...B_n$ for which specific service (G_k) the Owner will grant access using her credentials (C_x). The agreement is done at the user's discretion through any available out-of-band channel and is limited by the implemented technical capabilities of the system (i.e., for supported use cases implemented by DELEGATEE).
4) During the agreement, users exchange their unique identifiers (i.e., system username) so that the Owner from party A knows whom to authorize from party B.
5) The Owner A_i establishes a secure channel to the system, specifies for which credentials (C_x) she wants to perform the delegation, for which service (G_k) and to whom (username of B_j), while she additionally specifies the access control policy P_{ijxk} that restricts usage.
6) After receiving the confirmation, A_i disconnects.
7) The Delegatee B_j now establishes a secure channel to the system and can immediately see that she has been delegated credentials for a particular service. The credentials are hidden for the Delegatee B_j. If the Delegatee wants to access the service G_k, she may proceed.
8) The access to the service is always proxied through the

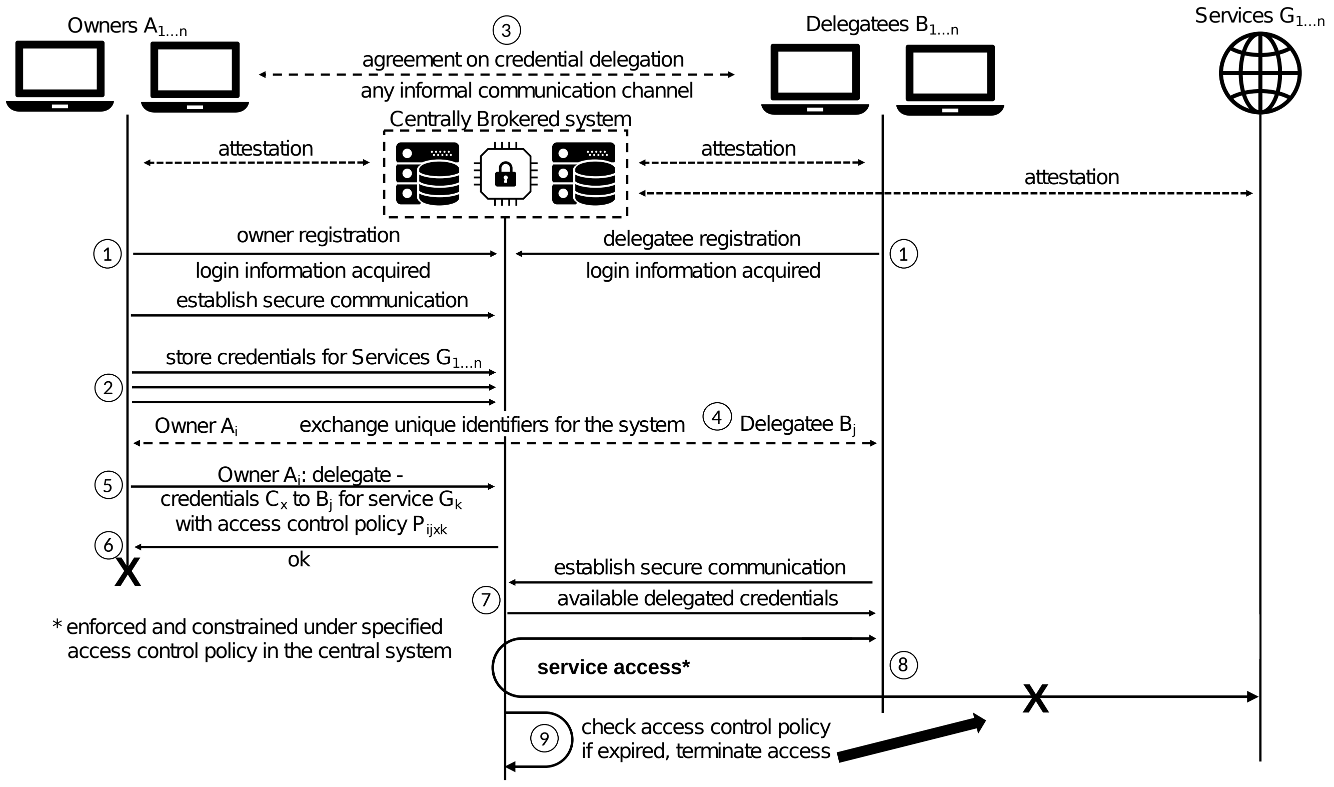
Owners A1...n
3
agreement on credential delegation
any informal communication channel
Delegatees B1...n
Services G1...n
Centrally Brokered system
attestation
attestation
attestation
1
owner registration
login information acquired
delegatee registration
login information acquired
1
establish secure communication
store credentials for Services G1...n
2
Owner Ai
exchange unique identifiers for the system
4
Delegatee Bj
5
Owner Ai: delegate -
credentials Cx to Bj for service Gk
with access control policy Pijxk
6
ok
X
establish secure communication
7
available delegated credentials
* enforced and constrained under specified
access control policy in the central system
service access*
8
X
9
check access control policy
if expired, terminate access

Fig. 1: DELEGATEE's system architecture

central broker with no direct communication between the Delegatee and the service. Any attempt to circumvent this results in protocol termination (e.g., if the user clicks an external link outside the proxied service).

9) After the defined access control policy expires (e.g., if it is time-limited) the Delegatee B_j loses access and the credentials are no longer delegated.

Authentication mechanisms. Note that attestation alone is not enough to authenticate the enclave and the Delegatee. The Owner must be sure that she connects to the correct enclave and that only the intended Delegatee is able to use the credentials.

IV. Security Analysis

In this brief security analysis, we mostly describe how Intel SGX protects the credentials, how it enforces access policies and how only the intended Delegatee is allowed to use the service. We consider a strong attacker that has hardware access and root platform privileges on the centrally brokered system as well as complete control over the network.

Remember that the processor encrypts the memory of an Intel SGX enclave with a key that is inferred from the platform embedded key and the measurement of the enclave amongst others. Therefore, any local attacker cannot just read out the runtime memory of an enclave. Since the memory of an enclave is kept in the clear only in cache, the processor protects the cache lines that belong to the enclave from any other software running on the platform. A malicious operating system also cannot access any cache entries of an enclave.

Intel SGX provides another important primitive: Attestation. Attestation allows any participant to verify the code that is running inside the enclave and that it is a genuine Intel SGX enclave. Before any interaction with the enclave, all participants first perform attestation and, only if it succeeds, they proceed. Therefore, they can verify that the policy is always enacted and that only the intended delegatee can access the delegated credentials.

We use TLS on all outgoing and incoming connections to protect against a network attacker. The attacker can, however, cut arbitrary connections. We consider this to be a similar attack vector to pulling the plug as it is essentially a Denial of Service which we consider out of scope.

V. Prototype Implementation

We implemented all four previously mentioned use cases, but we will only go into detail on two of them: mail/office and payments (PayPal). DELEGATEE also supports other use cases and we refer the reader to the full paper for a detailed description [13]. To realize the system architecture, we split it into two parts: the service-specific stateless enclaves and a management enclave. The management enclave, further referred to as API, authenticates the Owners and Delegatees and stores credentials. Users interact with the system through an ordinary website and a browser extension for increased usability. All communication between the users, the enclaves and the browser extension is done using TLS with replay protection. In this Section we presume that the Owner A_i and Delegatee B_j authenticate to the management enclave with

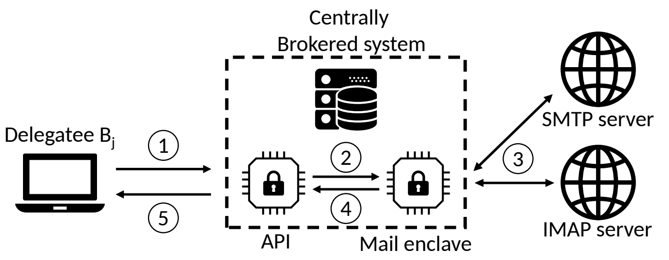

Fig. 2: Sending and receiving emails through DELEGATEE.

username and password and the Owner has already authorized the Delegatee by storing credentials C_x and access policy P_{ijxk} for a specific service. Thus, the Owner A_i is not shown in the figures.

The operating system is relied upon to handle incoming and outgoing TCP connections while the SSL endpoints reside in the trusted enclaves. We use the `mbedtls` library developed by ARM, which also comprises the bulk of our trusted computing base (TCB).

A. Mail/Office

Delegation of email accounts under a specific access policy, one of the DELEGATEE motivated applications, is implemented in the mail enclave. IMAP and SMTP clients are implemented to allow a Delegatee B_j to read and send emails using the delegated credentials C_x. Below we describe the architecture depicted in Figure 2:

1) The Delegatee B_j wants to use some credentials C_x that have been delegated by A_i. B_j connects securely to the centralized API using her username and password. She then requests to perform some action using C_x.
2) The API verifies that the Delegatee has access to C_x and then forwards the request, C_x and the corresponding policy P_{ijxk} to the mail enclave.
3) The mail enclave connects to either the SMTP server (for sending mail) or the IMAP server (for receiving mail) and executes the requested operation.
4) P_{ijxk} gets applied to the response from the external servers (IMAP) or to the outgoing requests (SMTP) and the resulting response gets forwarded to the API.
5) The API delivers the final response to B_j.

B. Payments

PayPal does not want to endorse giving away your credentials or automating the payments as this could compromise their security. Thus it is non-trivial to automate a PayPal payment and there is no public API. We must emulate a browser inside our enclave that accurately simulates a real user. Our implemented emulated browser follows redirects, fills known forms, and handles cookies until the final confirmation page is reached. The enclave then returns a confirmation `id` to the issuer that is used by the merchant to finalize the payment. Our browser extension simplifies the use of delegated PayPal credentials by adding a DELEGATEE checkout button next to the original PayPal checkout button if the Delegatee is logged in to our system and has some delegated credentials. Upon clicking on the DELEGATEE checkout, the Delegatee can choose one of the available PayPal credentials delegated to her and then the automated payment process starts. After that, no further user interaction is needed and the Delegatee will be forwarded to the confirmation page of the merchant if the payment succeeds. Below we describe the architecture depicted in Figure 3:

1) The Delegatee B_j wants to buy something from a merchant using credentials C_x delegated by A_i. B_j connects to the merchant and asks for a PayPal payment.
2) The merchant uses PayPal API to create a payment.
3) The payment is then forwarded to B_j.
4) B_j connects securely to the centralized API enclave using her username and password. She then requests to pay with PayPal using C_x.
5) The API enclave verifies that the user can access to C_x and then forwards the request, C_x and the corresponding policy P_{ijxk} to the PayPal enclave.
6) The PayPal enclave connects to PayPal and pays the payment with C_x if it is allowed by the policy P_{ijxk}. The PayPal service responds with a confirmation number.
7) The confirmation number is forwarded to the API.
8) The API delivers the confirmation number to B_j.
9) B_j forwards the confirmation number to the merchant and then the PayPal payment is finalized by the PayPal API using the received confirmation number.

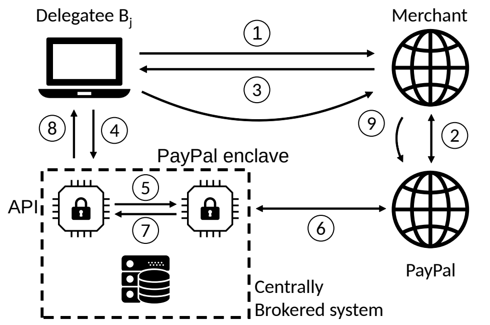

Fig. 3: PayPal payments through DELEGATEE.

Type	Test case	Mean (± std)
$a)$ SSL handshake	`openssl`	52.12ms (± 3.62)
	`mbedtls`	57.14ms (± 3.37)
	`mbedtls` in SGX	105.22ms (± 4.23)
$b)$ Mail	*normal operation*	1.12s (± 0.27)
	mail enclave	1.19s (± 0.22)
$c)$ PayPal	*normal operation*	25.92s (± 6.83)
	PayPal enclave	27.00s (± 4.35)

TABLE I: Latency for a) SSL handshakes, b) receiving e-mails in inbox, and c) executing PayPal transactions. Samplesize: 1000.

VI. PERFORMANCE ANALYSIS

Table I-a shows an overhead of around 50ms for a full SSL handshake using `mbedtls` inside an enclave compared to normal `mbedtls` and `openssl`. The handshake involves three exchanged messages, thus at least three `ocalls`/`ecalls`, all of which have to copy potentially large buffers. Previous work has shown that calls that copy large buffers across the trust boundary infer a large overhead. In our measurements, we recorded 19 `ocalls` during a request to the enclave.

The mail enclave incurs minimal overhead (Table I-b) with one extra handshake to the IMAP server. The overhead of this one additional handshake is insignificant compared to the time waiting for a response. In our test, we retrieve all emails from the account inbox.

The PayPal example does not seem to suffer from any delay added by our implementation (Table I-c). Note that we performed tests using the sandbox environment, provided by PayPal itself for testing integration with their services. This environment is feature-complete but slow as it is only functionality-oriented. Most time is spent by waiting for the PayPal servers. We also conducted tests in the real PayPal environment executing a real payment and buying an item online with a merchant supporting PayPal. However, due to the *CAPTCHA* protective mechanism involving the Delegatees' actions, it is not feasible to measure performance, since it depends on the user input.

VII. Discussion & Limitations

a) Authentication challenges.: Authentication in modern web services is complex. It can involve not just passwords but additional factors such as personal questions, email challenges, phone challenges, and "two-step authentication" apps such as Authy and Google Authenticator. Some of these can be supported with DelegaTEE, such as, email challenges or 2FA apps that could run inside the enclave as well, while for some, e.g., phone challenges, the current design of DelegaTEE cannot overcome the challenge.

Contextual factors, such as the IP address, time of day, and nature of service requests are often used to detect malicious or irregular behavior. Financial services, i.e., PayPal, have particularly sophisticated fraud detection regimes; e.g., ordering unusual products with Paypal may trigger a fraud alert. Consequently, the service might ask for additional authentication steps and a single credential in the form of a password may not suffice to delegate a resource or service via DelegaTEE. CAPTCHAs can easily be solved by forwarding them to the delegatee

To illustrate, consider a scenario where an Owner Alice (an inhabitant of the U.S.) delegates her password to DelegaTEE and allows her PayPal account to be rented. Suppose then that Delegatee Bob, in Nigeria, rents Alice's PayPal account in a prescribed way and attempts to execute a transaction. Paypal will see an unusual request coming from an IP address in a country with a different risk profile than the U.S., and potentially one that Alice has never visited. Bob's transaction request is likely to be suspicious. PayPal may then deny the transaction or request additional confirmation, e.g., via e-mail, to proceed. If Alice is unavailable or denies the transaction - which she may fail to recognize as originating with her delegation - the transaction will fail.

b) Authentication collisions.: Attempts at simultaneous use of a resource may fail, as many web services do not support multiple concurrent sessions for a given account. For example, if Alice has delegated use of her bank account to Bob, then she may be unable to use it herself while Bob (or DelegaTEE, to be precise) is logged in. Such collisions can be treated by invoking failure modes like those for basic authentication failures. Other policies are possible, however. For example, Owner Alice may set a policy that only delegates her resource at times when she is unlikely to use it. A small enhancement to DelegaTEE can also enable Alice to preempt the session of a Delegatee if desired.

c) Usability, Deployment and Service Prevention.: Throughout the paper we have presented multiple use-cases and implemented prototypes that support delegation of different services. The usability of these services by potential Delegatees is as if they were using the original service as its Owner. However, the usability of the DelegaTEE in general depends on the supported use-cases. A limitation of our system is that for each and every use-case a specific module (that matches the capabilities and technical challenges) has to be implemented. Until now, we have not found a way in order to develop a generic module that could support a wide variety of services. For example, interpreted languages, such as Javascript, remain an open problem since by executing unmeasured code in an enclave running the interpreter we cannot guarantee the security properties of DelegaTEE. In addition to that, almost all services (even the ones from the same category) have different user mechanisms, UI and control. Thus, a specific policy needs to be created that matches these controls in order to allow Owners to specify how their service could be used by potential Delegatees. Due to the complexity, the policies have to be created beforehand along with the implemented delegation scenario, while the end-user involvement is limited to configuring parameters, out of a set of given policy characteristics.

If all service operators would share a unique set of API calls that could cover the full functionality of their services, then the deployment of DelegaTEE would be feasible for almost all service categories. This would also allow for the creation of more general and richer access control policies that could be created by the end-users of the service as well, possibly overcoming the initially discussed complexity of complete policies that require serious engineering and evaluation of each specific use-case scenario.

However, it is hard to imagine that the service operators would view the above even as a viable option. In many cases, DelegaTEE allows the creation of secondary markets and poses a threat to the revenue stream of the original service operator. Additionally, DelegaTEE reduces the operators' ability to control and track their users since virtually, the number of users could grow but they would be seen only through the increased activity of users registered to the original service. Thus, most service operators would try to deny service access if executed through this form of delegation. As already mentioned, IP geofencing, pattern matching of actions and service usage, 2FA, along with the already existing fraud-detection mechanisms may endanger the functionality of our system. We stress that most of these protection mechanisms present a cat-and-mouse game between the service provider and DelegaTEE. In the security world, the attacker is usually a step ahead of the service provider since the provider has to react to the adversaries actions. In this case, however, the advantage lies with the service. Any change to the user interaction with the service requires a change in the service-specific enclave, therefore rendering it unusable for a short time.

d) Secondary markets.: Brokered delegation could give rise to offerings that compete directly with those of the very platforms hosting the delegated resources.

Facebook users could sell opportunities for "sponsored post" - unsolicited advertisements sent to their networks of friends or shown on their walls, as discussed above. Facebook users would then compete with Facebook itself in selling ads. Similarly, users could rent use of their Netflix account. Account sharing is already common within families and close friend circles. Brokered delegation could enable broad reselling and foster competition with direct sales of the subscription service.

Such secondary markets would in many cases violate providers' existing terms of service and might resemble markets for underground sales of virtual goods [12]. Those underground markets have met with two responses, sometimes used in tandem: (1) providers aim to detect facilitators of secondary markets and penalize or ban them, and (2) providers themselves seek to capture the revenue streams generated by secondary markets. DelegaTEE could provoke similar responses.

Peer-to-peer cryptocurrency-for-fiat exchanges is another setting that can benefit from DelegaTEE. Today, websites like LocalBitcoins.com receive Bitcoin deposits and hold them in escrow. Then they match-make and allow a buyer and a seller to negotiate an e-banking transfer. When the receiver gets the bank transfer, they instruct the LocalBitcoins service to complete the payment from the escrowed funds. If the receiver raises a dispute, then the service must investigate and ultimately determine whether to release the funds. However, such services naturally have limited investigative ability. They may call the user's bank, or ask both parties for evidence (i.e., screenshots). Neither option is satisfactory; the latter is prone to forgery, while the former may inadvertently draw suspicion to the user's bank account. Credential delegation provides an alternative, simplifying this business model and implementing a secure intermediary that guarantees execution and fair exchange.

VIII. Conclusion

The range of new sharing-economy applications we have considered as targets for DelegaTEE are just the tip of the iceberg. Many others—good, bad, and legally and morally ambiguous—await discovery: sharing of characters or virtual goods in online role-playing games, trustworthy delegation to financial advisors of online portfolio adjustment, safer rental of botnets by blackhat hackers, etc.

How brokered delegation may impact existing industries is an intriguing question. Service providers could attempt to detect and block services such as DelegaTEE. Considering such an adversarial model, would robust brokered delegation be possible? Alternatively, service providers could embrace the market disruption and co-opt users or central brokers as re-sellers of their services, turning a threat into an opportunity.

Other important questions arise regarding user privacy, rights, and control of personal information. If users can delegate control selectively, would they more commonly outsource curation of their social media personas and e-mail communications, in effect creating personal information supply chains? Conversely, would such a system empower employers to be more aggressive in monitoring and controlling their employees' social media accounts, all with the excuse of being able to tailor safeguards via brokered delegation?

Whatever happens, we believe that brokered delegation will create a crop of new and more pervasive resource-sharing models, further advancing the proven disruptive power of sharing economies.

References

[1] Alexander, B. Introduction to Intel SGX Sealing, 2016. https://software.intel.com/en-us/blogs/2016/05/04/introduction-to-intel-sgx-sealing.

[2] Alves, T., and Felton, D. TrustZone: Integrated Hardware and Software Security-Enabling Trusted Computing in Embedded Systems, 2004.

[3] Bohannon, J. Who's downloading pirated papers? Everyone, 2016.

[4] Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., and Sadeghi, A. Software Grand Exposure: SGX Cache Attacks are Practical, 2017. http://arxiv.org/abs/1702.07521.

[5] Checkoway, S., and Shacham, H. Iago attacks: Why the system call api is a bad untrusted rpc interface. In *ASPLOS* (2013), vol. 13, pp. 253–264.

[6] Costan, V., and Devadas, S. Intel SGX explained. In *Cryptology ePrint Archive* (2016).

[7] Costan, V., Lebedev, I. A., and Devadas, S. Sanctum: Minimal hardware extensions for strong software isolation. In *USENIX Security Symposium* (2016), pp. 857–874.

[8] Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., Feldman, A. J., Appelbaum, J., and Felten, E. W. Lest we remember: Cold-boot Attacks on Encryption Keys. *Communications of the ACM 52*, 5 (2009), 91–98.

[9] Johnson, S., Scarlata, V., Rozas, C., Brickell, E., and McKeen, F. Intel SGX: EPID provisioning and attestation services, 2016. https://software.intel.com/en-us/blogs/2016/03/09/intel-sgx-epid-provisioning-and-attestation.

[10] Kauer, B. OSLO: Improving the Security of Trusted Computing. In *USENIX Security Symposium* (2007), pp. 229–237.

[11] Lee, D., Kohlbrenner, D., Karandikar, S., Ou, A., Asanovic, K., Song, D., Lebedev, I., and Devadas, S. Keystone - Open-source Secure Hardware Enclave. https://keystone-enclave.org/, 2018.

[12] Lehdonvirta, V., and Castronova, E. *Virtual economies: Design and analysis.* MIT Press, 2014.

[13] Matetic, S., Schneider, M., Miller, A., Juels, A., and Capkun, S. Delegatee: Brokered delegation using trusted execution environments. In *USENIX Security Symposium* (2018), pp. 1387–1403.

[14] McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., and Savagaonkar, U. R. Innovative instructions and software model for isolated execution. In *HASP@ ISCA* (2013).

[15] Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T. F., Yarom, Y., and Strackx, R. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *27th {USENIX} Security Symposium ({USENIX} Security 18)* (2018), pp. 991–1008.

[16] Wojtczuk, R., and Rutkowska, J. Attacking SMM memory via Intel CPU cache poisoning. *Invisible Things Lab* (2009).