Verifiable and Auditable Digital Interchange Framework[#]

Prabal Banerjee[†], Dushyant Behl[*], Palanivel Kodeswaran[*], Chaitanya Kumar[*], Sushmita Ruj[‡] and Sayandeep Sen[*]

[†]Indian Statistical Institute Kolkata, [*]IBM Research, [‡] CSIRO Data61 Australia and Indian Statistical Institute Kolkata

ABSTRACT

We address the problem of fairness and transparency in online marketplaces selling digital content, where all parties are not actively participating in the trade. We present the design, implementation and evaluation of VADER, a highly scalable solution for multi-party fair digital exchange that combines the trusted execution of blockchains with intelligent protocol design and incentivization schemes. We prototype VADER on Hyperledger Fabric and extensively evaluate our system on a realistic testbed spanning five public cloud datacenters, spread across four continents. Our results demonstrate that VADER adds only minimal overhead of 16% in median case compared to a baseline solution, while significantly outperforming a naive blockchain based solution that adds an overhead of 764%.

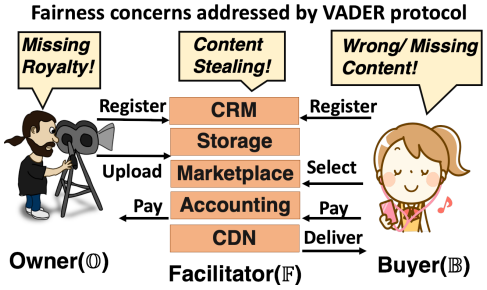


Figure 1: Present day video exchange

1 INTRODUCTION

Online media consumption is a big business [13], with users watching billions of hours of videos per month [90] and media traffic constituting roughly 70% of downstream internet traffic [80].

A key reason for this success, lies in the simplicity of present day online media (and money) exchange process as depicted in Fig. 1. As shown, a present day content creator can simply upload content and get paid based on viewership (or sales). Similarly, buyers can pay the right price to access media without worrying about content authenticity, price gouging, non delivery etc. The ease of operation is due to presence of facilitators such as Youtube, Netlflix, iCloud etc. As shown in Fig. 1, the facilitator provides all the ancillary but critical services of content hosting, searching, delivery, payments etc. to complete the digital ecosystem for online media consumption. While the efficacy of present day media delivery systems is reflected in their success, they suffer from an important shortcoming in the *inability to guarantee honest behavior* and vulnerability to fraudulent behavior by participants. This is due to the fact that present day systems lack the underpinnings to demonstrably guarantee honest behavior; forcing both buyers and owners to *trust* that a facilitator behaves in a *fair* manner.

For example in Fig. 1, the owner trusts the facilitator to honestly report the number of views/-downloads and calculate its royalty payments in a transparent manner. Similarly, the buyer trusts the facilitator to deliver the right content against payment. Finally, the facilitator expects the buyer to pay for a successful delivery, without falsely alleging non-delivery.

However, lack of baked-in guarantees of honest behavior can lead to disputes, such as a buyer fraudulently alleging non-receipt of content and denying payments thus stealing content from facilitator; facilitator mis-reporting sales to cheat owner of dues, or even charging buyers without providing right content.

In fact, recent events highlight the inadequacy of this trust based model [17, 19, 86]. Specifically, content owners have raised a number of complaints against faclitators regarding their royalty earnings and lack of clarity in the calculation [10, 18, 67], highlighting discrepancies in earnings with reported

Authors are listed in alphabetical order.

viewership. Similarly, buyers have also raised disputes regarding content received from these platforms [2].

We believe that such disputes can only increase in future to the detriment of the growth of online media delivery platforms. This in turn, motivates us to address this important problem of *guaranteeing* successful video exchange amongst mutually untrusting buyers and facilitators. We ensure each party (owner, buyer and facilitator) gets their rightful outcome or no one does. We define the above as **Multi-party fair digital exchange** (abbreviated as *fair exchange* in the rest of the paper) among the owners, buyers and facilitators in the marketplace model.

Note that prior work [30, 48, 49] for providing fairness are not applicable in the above setting as they *trust the facilitator* to grant access to internal logs and metrics, an important assumption we intend to avoid. Similarly, decentralized video delivery platforms [32, 40, 62, 64, 88] which bypass centralized facilitators (thus obviating the need to *trust* them), while promising are not suitable. This is due to the fact that decentralized content delivery platforms suffer from poor content quality, sporadic availability, rampant illegal content etc., ironically due to absence of a facilitator's dedicated resources in maintaining the platform [20].

Motivated by above observations, we set our goal to develop a *readily usable* solution for *fair exchange*, which would be a)*compatible and incrementally deployable with present day facilitator driven marketplace systems* and b) *should be able to provide transparency and fairness into existing video delivery platforms with minimal overhead in terms of modifications and performance.*

In this paper we present the system design, implementation and evaluation of Verifiable and Auditable Digital Interchange Framework (VADER) which satisfies the above mentioned criteria.

In process of designing VADER, we study various fraud risks that arise in the marketplace model and note that guaranteeing multi-party *fair exchange* in this model presents a unique challenge. As shown in Fig. 1, the owner is a *passive party*, in the sense that after video upload, it is not directly involved in the exchange of video and money between the *active parties* viz. the facilitator and buyer. Being a passive party, an owner is completely dependent on the honesty of facilitator, as it has no way to learn of exchanges of its content being done. This makes the owner vulnerable to being misled about its true earnings, either by the facilitator [10] alone, or in collusion with the buyer [2, 9]. Fig 1, highlights the specific risks faced by the respective parties.

We note that while the problem of fair exchange among active parties is well studied in theory [47, 71, 78, 89]; to the best of our knowledge, protecting rights of passive parties, without significantly altering the flow of video exchange [1] *is a new paradigm for fair exchange, not yet covered in literature.*

In VADER we not only protect the buyer and facilitators against various *active party risks* but secure owner's interest from *passive party risks*. To the best of our knowledge, VADER is the first real system to demonstrate such capabilities.

VADER accomplishes low overhead *fair exchange* solution by leveraging the following key insights,
→**Insight 1)** We can *guarantee fair exchange* by sending encrypted video and performing fair exchange of only the *key* and money. This enables us to leverage the existing optimized delivery infrastructures of facilitators for sending (encrypted) content and making system for fairness incrementally deployable.
→**Insight 2)** By assuming the presence of a trusted arbitrator that is slow (when compared to direct interaction without intermediary) but can deterministically detect a malicious party and provide restitution (right encrypted content, key or money) to the honest party, parties can opportunistically exchange key and money directly between themselves without having to interact with the arbitrator unless there is a dispute.
→**Insight 3)** Assuming parties are rational, introduction of bounties (that are large and funded by penalizing malicious parties) for reporting misbehavior introduces an element of distrust between parties, thus preventing collusion aimed at subverting the protocol. Note that, the first two insights enable efficient operation, guaranteeing fairness for the active parties; while the third insight ensures fairness for the passive party under assumption of rational participants.

We select blockchain as the tamper-proof ledger and execution platform for VADER. Our selection of blockchain is motivated by the fact that it offers decentralization of trust and auditability guarantees sought by VADER. Furthermore, the native blockchain cyptocurrency can be used to design incentivization schemes and programmatically enforce desired behavior from the interacting parties as mandated by our insights above. We also

[1] i.e., without making owner also an active party by say asking for its approval on every trade

point out the second insight of opportunistic exchanges on fast path (batching), while reverting to the slow but guaranteed path is also used in state-channels for scaling transaction throughput. However, these solutions involve native assets such as cryptocurrencies giving complete control to the arbitrator to revert back the state (e.g. ownership) of an asset. On the other hand, we deal with non-native assets such as decryption keys which once delivered to the buyer are unaffected by blockchain asset state. We modify the state-channel protocol to account for the above oddity, as described in Sec. 3.1.

As part of this work, we have systematically studied exchange process in present day video delivery platforms and used the insights to design and implement VADER. VADER protocol carefully combines insights from diverse domains of cryptography, incentives design, blockchains to ensure *fair exchange* for video exchange. VADER is incrementally deployble with minimal modifications to present day video delivery platforms. Specifically, in Sec. 3 we design VADER protocol (message exchange sequence) and perform a comprehensive security analysis to show how VADER protects honest parties against attacks in Sec. 3.4. We implement VADER incorporating all the insights described above and extensively study the performance of VADER over two baseline techniques, through extensive evaluation across realistic workloads.

We note that while we use video as an example digital asset in this paper, VADER works for any digital content [5, 6, 11, 12] that can be provably verified, say using cryptographic digests. Additionally, the owner in VADER is a logical abstraction that can represent multiple entities that need to be paid royalties individually, as in the music industry [67]. Finally, VADER focuses on guaranteeing *fair exchange* between the buyer, facilitator and owner. VADER does not address the complementary issue of content piracy, where a buyer buys content legally on a VADER based marketplace and then resells it with minor modifications. Other works [15, 81], including recent work [66] use a combination of content watermarking and on-blockchain penalty mechanisms to prevent content piracy which can be easily embedded into VADER smart contracts.

We believe that *fair exchange* systems will be the norm in near future and our work advances the nascent area of designing practical systems for Multi-Party Fair exchange. We claim the following as key **contributions: 1)** To the best of our knowledge, we are the first to formulate and present the problem of *multi-party fair digital exchange* in third party marketplace scenario where one of the parties is passive and does not directly interact with the buyer (Sec. 1 & 2). **2)** We design the VADER protocol and study its security properties. **3)** We implement VADER protocol on Hyperledger Fabric, and extensively evaluate its performance on a realistic test-bed of upto 91 nodes spread over 4 continents, transferring at least 50TB of data over the network. We find that VADER adds only minimal overhead of 16% in median case compared to the baseline VANILLA solution.

Outline: The rest of the paper is organized as follows. In Sec. 2, we formally describe the problem and the solution requirements. In Sec. 3, we describe our solution and discuss the security analysis of our work in Sec. 3.4. We present details of VADER implementation in Sec. 4 and in Sec.5, we present performance evaluation of VADER under realistic conditions. In Sec. 6, we discuss the related work and finally conclude our paper in Sec. 7

2 PROBLEM SETUP

We define Owner ($\mathbb{O}$) as a party, such as artists or financiers, that need to be monetarily compensated for every successful trade/download of the digital content. We define Facilitator($\mathbb{F}$) as a party responsible for conducting the trade on behalf of $\mathbb{O}$. Finally, we define a Buyer($\mathbb{B}$) as a party that is buying digital content from $\mathbb{F}$'s marketplace.

As explained in Sec. 1, $\mathbb{O}$ is the *passive party* while $\mathbb{B}$ and $\mathbb{F}$ are *active parties* in the digital exchange. Further, we assume the parties *do not completely trust each other* and $\mathbb{B}$ and $\mathbb{O}$ do not know each other. We also assume that the parties are rational and will collude with each other to maximize profit.

A **Multi-party Fair Digital Exchange** protocol in the above context, should ensure that either *all* parties receive their desired item in exchange for their own item/service, or *none* of the parties receive anything. Specifically, **Multi-party Fair Digital Exchange** guarantees the following behavior: **a)** A $\mathbb{B}$ will always get the correct video uploaded by $\mathbb{O}$ if it can submit proof of payment **b)** A $\mathbb{F}$ will always get paid if it can submit evidence of sending the correct video **c)** An $\mathbb{O}$ will always get paid if $\mathbb{F}$ was paid by selling its content.

Attacks: A *fair exchange* protocol for video delivery must protect against the following attacks: **Atk.1- Royalty Manipulation:** constitutes any

manipulation in the royalty calculation of $\mathbb{O}$ e.g. a malicious $\mathbb{F}$ can cheat $\mathbb{O}$ by a) misreporting the number of downloads or b) selling the content to $\mathbb{B}$ at a higher price while compensating $\mathbb{O}$ with a lower royalty by exploiting the information asymmetry between $\mathbb{B}$ and $\mathbb{O}$. **Atk.2- Content Mismatch**: constitutes delivery of content to a $\mathbb{B}$ that is different from the one promised while receiving money. **Atk.3- Content Stealing**: constitutes gaining access to the content without compensating the designated parties for their fair share of payment for the same, e.g. $\mathbb{B}$ getting access to content without paying money to $\mathbb{F}$ by claiming to have received wrong content despite getting right content, thus cheating both $\mathbb{F}$ and $\mathbb{O}$.

Under our threat model, the above attacks represent a completely exhaustive list of attacks on marketplace fairness. We reiterate that VADER does not address the complementary issue of content piracy, and is compatible with other solutions [15, 66, 81] that address piracy.

3 SOLUTION DESCRIPTION

We describe VADER protocol by presenting an outline of the solution in Sec. 3.1 and the details of various protocol steps in Sec. 3.2. We also, present a security analysis of our protocol in Sec. 3.4.

• **Notation:** We use the following notation in the description of our solution. Let λ be a security parameter, $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ be a cryptographically secure hash function, $(Gen(\lambda), Enc(k,m), Dec(k,m))$ be a IND-CCA secure symmetric encryption scheme and $(KeyGen(\lambda), Sign(sk,m), Verify(pk,m,\sigma))$ be a public key cryptosystem based secure digital signature scheme. We assume that at bootstrap, each party generates a key pair (pk, sk) using $KeyGen$. We use $\mathcal{L}$ to represent the distributed blockchain ledger and ζ to represent a state channel between two parties.

3.1 Solution Outline

VADER aims to deliver an efficient solution for guaranteeing *fair exchange* by leveraging the three insights described in Sec. 1. Following the first insight, $\mathbb{F}$ utilizes its usual content delivery infrastructure to send encrypted video to $\mathbb{B}$ thereby reducing the problem of fair video exchange to that of fair exchange of key and money.

Fig. 2 depicts the phases of video exchange using VADER. In the Initialization phase, leveraging the second insight, $\mathbb{F}$ and $\mathbb{B}$ lock money in the blockchain escrow to create a channel, akin to state-channels [42, 74]. We define *channel* as a unique bi-directional message queue between two parties, backed with blockchain escrowed money, enabling parties to conduct multiple exchanges directly without hitting blockchain (Channel construction and lifecycle described later). This money is then used as collateral in the Exchange and Finalization phases where the $\mathbb{B}$ and $\mathbb{F}$ exchange

Algorithm 1 Conditional Escrow Contract

$IOU :=< sender, recvr, \$amt >$

function OpenEscrow($B, amt, \sigma_B, \tau, cond$)
  if $Verify(pk_B, amt, \sigma_B) = true$ **then**
    $e_{id} \leftarrow genEscrow()$
    $e_{id}.[owner, bal, timeout, cond] \leftarrow$
      $[B, amt, \tau, cond]$
    return e_{id}
function $ProcessIOU([IOU(s)], evidence)$
  if $block_{ht} \leq e_{id}.timeout$ AND
    $Verify(evidence, e_{id}.condition)$ **then**
  for each $iouinIOU(s)$ **do**
    bal -= $iou.amt$
    $send(iou.recvr, iou.amt)$
function $CloseEscrow(e_{id})$
  if $block_{ht} \geq timeout$ **then**
    $send(e_{id}.owner, e_{id}.balance)$

Algorithm 2 State Channel Contract

τ represents the settlement timeout

function Open($\mathbb{B}$,B_{amt},σ_B,$\mathbb{F}$,F_{amt},σ_F,τ)
  ζ= $GenChannel()$
  $\zeta.cid \leftarrow \{0,1\}^k$
  $\zeta.timer\_started = false$
  $\zeta.\mathbb{B}.escrow = OpenEscrow(\mathbb{B}\, B_{amt}, \sigma_B, \tau)$
  $\zeta.\mathbb{F}.escrow = OpenEscrow(\mathbb{F}\, F_{amt}, \sigma_F, \tau)$
  $\zeta.timeout$, $\zeta.state = \tau$, $Open$
  $\mathcal{L}$.Store(ζ)
function Close(ζ,$ChanUpdateMsgs$)
  if $\zeta.timer\_started$ AND $block_{ht} \geq \zeta.timeout$
    return
  for each $m \in ChannelUpdateMessagese$ **do**
    $VerifyValidity(m)$
    $iou = m.extractIOU()$
    $ProcessIOU(\zeta.escrow(s), iou)$
  $\mathcal{L}$.Store($\zeta.timer\_started = true$)
function settleChannel(c_{id})
  if $\zeta.timer\_started$ AND $block_{ht} \geq \zeta.timeout$
    $CloseEscrow(\zeta.escrow(s))$
  $\mathcal{L}$.Store($\zeta.state = Closed$)

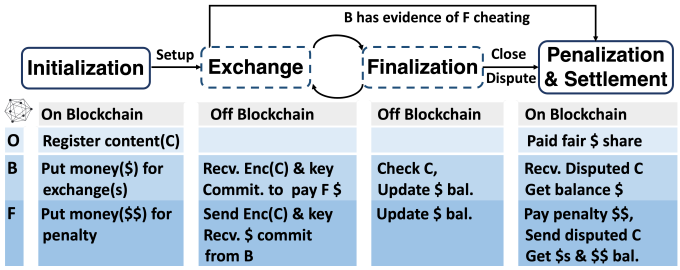


Figure 2: Video exchange lifecycle in VADER

ID	Content	Direction	Intent
M_0	$<c_{id}, req_{id}, id_C, price_{idC}, \sigma_B>$	$\mathbb{B} \rightarrow \mathbb{F}$	$\mathbb{B}$ Content Request
M_1	$<M_0, \sigma_F>$	$\mathbb{F} \rightarrow \mathbb{B}$	ACK M_0
M_2	$<E(c), id_E, \sigma_F>$	$\mathbb{F} \rightarrow \mathbb{B}$	Enc. Video + Digest
M_3	$<M_2, \sigma_B>$	$\mathbb{B} \rightarrow \mathbb{F}$	Enc. Video ACK
M_4	$<IOU, \sigma_B>$	$\mathbb{B} \rightarrow \mathbb{F}$	Optimistic IOU
M_5	$<k, \sigma_F>$	$\mathbb{F} \rightarrow \mathbb{B}$	Key release

Figure 3: VADER message exchanges

money and video directly offchain. Finally, in the Penalization and Settlement phase, disputes, if any, are resolved deterministically by the trusted arbitrator on-chain as shown in Fig. 2 and escrow money appropriately redistributed to $\mathbb{B}$ and $\mathbb{F}$. We note that, as shown in Fig. 2 only Initialization and Settlement phases interact with blockchain, while exchange and finalization phases are executed directly between $\mathbb{B}$ and $\mathbb{F}$ without hitting the blockchain. Moreover, $\mathbb{O}$ being the passive party does not have any actions during the exchange and finalization steps of the protocol.

VADER leverages the trusted execution of a suite of blockchain deployed smart contracts, to emulate a *trusted arbitrator*. The trusted arbitrator (in settle phase) works as follows: **1)** Any party can close the channel either due to dispute or logical end of application exchange, and submit evidence of protocol compliant behavior to the smart contract, if any **2)** VADER allows the counterparty to submit counterfactual evidence within a predetermined time period to state its view of the offchain protocol state. As shown in Fig. 2, the VADER smart contracts evaluate the evidence submitted by both parties and then either transfer money to $\mathbb{B}$ (refund) or $\mathbb{F}$ (payment) from an escrow that is funded in the initialization phase.

Finally, motivated by the third design insight, VADER protocol design introduces a bounty scheme that rewards $\mathbb{B}$ with a much larger monetary benefit, by penalizing $\mathbb{F}$ (compared to cost of the video) if it can submit evidence of cheating by $\mathbb{F}$ to the arbitrator. As mentioned before, under assumptions of rationality, the penalty scheme prevents collusion between $\mathbb{F}$ and $\mathbb{B}$ in which they both agree to alter the offchain message exchanges between themselves for mutual benefit (by dividing $\mathbb{O}$'s share between themselves), and submit the altered sequence in the settlement phase, thereby cheating $\mathbb{O}$ of its fair share. By making it more beneficial for $\mathbb{B}$ to report $\mathbb{F}$ and collect bounty, in turn, forcing $\mathbb{F}$ to honestly report exchange information to the arbitrator, VADER ensures the passive party, $\mathbb{O}$, gets its rightful share. Finally, we note that VADER uses two well known constructs described below.

Evidence of protocol compliance: VADER leverages cryptographic commitment schemes and digital signatures over messages exchanged between $\mathbb{F}$ and $\mathbb{B}$ as *evidence* of protocol compliant behavior by blockchain based trusted arbitrator.

Conditional Escrow: This construct allows parties to prove solvency to each other as the first step of digital exchange by locking money on blockchain and guaranteeing that the money cannot be unilaterally withdrawn by either party. This construct is used by the *trusted arbitrator* to lock money (crypto-currency) for a duration (τ) in the initialization phase and deterministically release money (IOU) based on submission of evidence ($cond$) for VADER protocol compliant operation in settle phase. Alg. 1, shows a conditional escrow contract, wherein an entity can lock money(amt), for a duration of time(τ). Money(IOU) is releasable before τ only on successful evaluation of condition $cond$. Alg. 1 in Appendix shows pseudocode for conditional escrow contract.

Our state channel construction builds on top of the conditional escrow contract. In Alg. 2 we provide our state channel algorithm providing open, close and settle semantics. In addition to the one's in Alg. 2 our state channel supports the following properties (used in subsequent algorithms), **1)** ζ.Store and ζ.Load:- which is the per party local channel storage where each party can save messages, **2)** ζ.Send and ζ.Recv:- using which one party can send a message to the other party involved in the state channel.

3.2 Solution Description

As mentioned in Sec. 3.1, VADER proceeds in the four phases as depicted in Fig. 2. We describe the specifics of each of the phases next.

Algorithm 3 Content Registration

amt_O ▷ % Royalty $\mathbb{O}$ should receive per Xchg

function $\mathbb{O}$.UpContent(C, n, amt_O)
  $c_1, c_2, \ldots, c_n = C$
  $id_C = \{H(c_1), H(c_2), \ldots, H(c_n)\}$
  $m =< id_C, amt_O, pk_O, pk_F >$
  Send <C, m, $\sigma_O = Sign(sk_O,m)$> to $\mathbb{F}$
  Receive $< Ack, vid >$ from $\mathbb{F}$

function $\mathbb{F}$.RecvContent($\mathcal{L}$)
  On Recv $< C, m, \sigma_O >$ from $\mathbb{O}$
  if $Verify(pk_O, m, \sigma_O) \neq true$ **then**
    terminate
  $id'_C = \{H(c_1), H(c_2), \ldots, H(c_n)\}$
  $< id_C, amt_O, pk_O, pk_F >= m$
  if $(m.id_C \neq id'_C)$ **then**
    terminate
  $vid = H(H(c_1)|H(c_2)|\ldots|H(c_n))$
  $\sigma_F = Sign(sk_F, < m, \sigma_O >)$
  $\mathcal{L}.Store(vid, m, \sigma_O, \sigma_F)$
  Send $< Ack, vid >$ to $\mathbb{O}$

Algorithm 4 Trade Agreement

id_C ▷ The content which $\mathbb{B}$ is requesting
$price_{id_C}$ ▷ Bid price $\mathbb{B}$ wants to pay
$cost_{id_C}$ ▷ Ask price which $\mathbb{F}$ wants

function $\mathbb{B}$.ContentReq(ζ, $id_C, price_{id_C}$)
  $reqid \leftarrow \{0,1\}^k$;
  $m =< \zeta.cid, reqid, id_C, price_{id_C} >$
  $\sigma_B = Sign(sk_B, m)$;
  $\zeta.Send(M_0 < m, \sigma_B >)$ to $\mathbb{F}$;
  ▷ M_1 is generated in ServContentReq
  $M_1 < m, \sigma_B, \sigma_F > = \zeta.Recv()$ from $\mathbb{F}$
  $\zeta.Store(M_0, M_1)$

function $\mathbb{F}$.ServContentReq(ζ, $\mathbb{F}$, $\mathbb{B}$)
  $M_0 < m, \sigma_B > = \zeta.Recv()$ from $\mathbb{B}$
  if $Verify(pk_B, m, \sigma_B) \neq true$ **then**
    $\zeta.Close()$
  $< reqid, id_C, price_{id_C} >= m$
  if $(price_{id_C} \neq cost_{id_C})$ **then**
    $\zeta.Close()$
  if ($reqid$ is prev. known in ζ) **then**
    $\zeta.Close()$
  $\sigma_F = Sign(sk_F, (m, \sigma_B))$
  $M_1 =< m, \sigma_B, \sigma_F >$
  $\zeta.Send(M_1)$ to $\mathbb{B}$
  $\zeta.Store(M_0, M_1)$

• **Phase 1: Initialization:** This phase involves $\mathbb{O}$ uploading content to the $\mathbb{F}$ and $\mathbb{B}$ setting up a channel to be used for multiple exchanges with $\mathbb{F}$.

Init.1 $\mathbb{O}$ - $\mathbb{F}$ **Content Registration:** As shown in Alg 3, Owners register their content for sale in the marketplace by uploading the content $C(=\{c_{1..n}\})$ composed of chunks $c_1, \ldots c_n$ along with its digest $id_C=\{H(c_1),\ldots,H(c_n)\}$ and amt_O, the percentage to be given to $\mathbb{O}$ for each sale of C (for privacy reasons amt_O is encrypted with the public key of $\mathbb{F}$). On receiving C, $\mathbb{F}$ verifies the message signature and records $\mathbb{O}$ as the unique owner and $\mathbb{F}$ as an authorized distributor on blockchain. Note that content registration is a *one time* operation between the $\mathbb{O}$ and $\mathbb{F}$. At the end of this step, the rightful ownership of video content is established on blockchain.

Init.2 $\mathbb{B}$ - $\mathbb{F}$ **Channel Creation:** During initialization, $\mathbb{B}$ funds the channel (with $\mathbb{F}$) with cryptocurrency (sufficient for multiple video exchanges) by locking money in an on-chain escrow. Similarly, $\mathbb{F}$ also makes an initial deposit, that is much larger compared to $\mathbb{B}$ś deposit (to cover penalties described later). Once both deposits are committed to blockchain, the state channel contract generates a unique channel id *cid* that is used to bind subsequent message exchanges between $\mathbb{B}$ and $\mathbb{F}$ to the channel.

• **Phase 2: Exchange:** The exchange phase is the core of VADER, as shown in Fig.2, where $\mathbb{F}$ and $\mathbb{B}$ perform exchange of the digital content. Fig. 3 shows the messages used in the construction of VADER exchange protocol as described below.

Xchg.1 Exchange Agreement: The goal in this step is to ensure that both $\mathbb{B}$ and $\mathbb{F}$ mutually agree on the video to be exchanged and the price to be paid by $\mathbb{B}$ to $\mathbb{F}$. As depicted in Alg 4, $\mathbb{B}$ submits to $\mathbb{F}$ the *Exchange Request message*, M_0, (m=<*cid*, *reqid*, id_C, $price_{id_C}$>, σ_B), containing the channel id *cid*, video identifier id_C, a *randomly generated, strictly monotonically increasing reqid*, as well as the amount $price_{id_C}$ that $\mathbb{B}$ agrees to transfer on successful exchange. On receiving M_0, $\mathbb{F}$ checks that the *reqid is indeed monotonically increasing, to ensure reqid is not reused (explained later in penalization scheme)*, and the transfer price is agreeable. In return, $\mathbb{F}$ sends counter-signed message M_1 = <M_0,σ_F> to $\mathbb{B}$, committing to the terms.

Xchg.2 Out-of-Band Encrypted Video Transfer: Next, based on insight 1, as depicted in Alg 5 $\mathbb{F}$ randomly samples key k and sends encrypted version of the requested video along with signed hashes of each individual encrypted chunk $id_E=\{H(E^k(c_1 \ldots c_n))\}$ in M_2 to $\mathbb{B}$.

Algorithm 5 Offline Video Exchange & $\mathbb{B}$ Ack

t ▷ Timeout $\mathbb{B}$ uses waiting for enc. content

function $\mathbb{F}$.SendContent($\zeta, C, reqid$)
  $k = Gen(\lambda)$
  $e_1, \ldots, e_n = Enc^k(c_1), \ldots, Enc^k(c_n)$
  $id_E = \{H(e_1), \ldots, H(e_n)\}$
  $\zeta.Send(< E =< e_1, \ldots, e_n >, id_E >)$ to $\mathbb{B}$
  $M_2 < id_E, \sigma_B > = \zeta.Recv()$ from $\mathbb{B}$
  $\sigma_F = Sign(sk_F, id_E)$
  $M_3 =< id_E, \sigma_F, \sigma_B >$
  $\zeta.Send(M_3)$ to $\mathbb{B}$
  $\zeta.Store(M_2, M_3)$

function $\mathbb{B}$.RecvContent($\zeta, reqid, t$)
  $< E, id_E >= \zeta.Recv()$ from $\mathbb{F}$
  Save $E =< e_1, \ldots, e_n >$
  $id'_E = \{H(e_1), \ldots, H(e_n)\}$
  for each i *in* id_E **do**
    if $id'_E[i] \neq id_E[i]$ **then**
      Request $\mathbb{F}$ for correct $e_i, id_E[i]$
      Start timer t; On Timeout:
        if correct chunk is received within t:
          Update $e_i, id_E[i]$
        else
          ▷ Matching chunk & hash is not received
          $\zeta.Close()$
  $\sigma_B = Sign(sk_B, id_E)$
  $\zeta.Send(M_2 < id_E, \sigma_B >)$ to $\mathbb{F}$
  ▷ M_3 is generated in SendContent
  $M_3 < id_E, \sigma_F, \sigma_B > = \zeta.Recv()$ from $\mathbb{F}$
  $\zeta.Store(M_2, M_3)$

On receiving encrypted video, $\mathbb{B}$ verifies if the digest matches with the one sent by $\mathbb{F}$ in M_2. In case of a match, $\mathbb{B}$ sends counter signed *Encrypted Video Acknowledgement* in M_3=<M_2, σ_B>. In case of a digest mismatch, $\mathbb{B}$ requests $\mathbb{F}$ for retransmission of specific chunks until either agreement is reached or failure after certain number of retries. In the next step, $\mathbb{B}$ and $\mathbb{F}$ exchange key and money in a trustworthy manner.

Xchg.3 Money-Key Exchange: In this step, as shown in Alg 6, $\mathbb{B}$ **1)** sends an *optimistic IOU* message M_4 to $\mathbb{F}$, and **2)** starts a timer t within which $\mathbb{F}$ needs to send the key k. In case of timeout, $\mathbb{B}$ closes the channel and initiates dispute resolution. On receiving the IOU, $\mathbb{F}$ releases the key along with digest in M_5.

• **Phase 3: Finalization:** In this phase, $\mathbb{B}$ verifies whether it received the right content or raises a dispute.

Final.1 Verification: As shown in Alg 7, $\mathbb{B}$ decrypts the video using k and verifies that $\{H(Dec^k(c_1 \ldots c_n)\}$ matches id_C uploaded by $\mathbb{O}$. In case of match, $\mathbb{B}$ has received desired content and the same channel can be re-used for future exchanges. However, in case of a mismatch, i.e. $\mathbb{F}$ sent wrong content, $\mathbb{B}$ closes the channel and registers a dispute with the *Dispute Resolve* smart-contract by submitting $M =< M_1, M_3, M_4, M_5 >$ as evidence along with at least one of the mismatched chunks, c_i.

Final.2 Channel Balance Update: At the end of a successful exchange, each party locally updates the channel balance of the counterparty and decides if the channel has sufficient balance to continue or must be closed.

Algorithm 6 Exchange

t ▷ Timeout $\mathbb{B}$ uses waiting for k

function $\mathbb{B}$.SendIOU(ζ, $reqid, price_{id_C}, t$)
  $IOU =< I = pk_B, OU = pk_F, price_{id_C} >$
  $\sigma_B = Sign(sk_B, IOU)$
  $\zeta.Send(M_4 < IOU, \sigma_B >)to\ \mathbb{F}$:
    Start timer t; On Timeout:
      if k is not received **then**
        $\zeta.Close()$
  $M_5 < k, \sigma_F >= \zeta.Recv()$ from $\mathbb{F}$
  $\mathbb{B}.DecryptAndVerify(\zeta, reqid, k, \sigma_F)$
  $\zeta.Store(M_4, M_5)$

function $\mathbb{F}$.SendKey(ζ, $reqid, k$)
  $M_4 < IOU, \sigma_B > = \zeta.Recv()$ from $\mathbb{B}$
  if $Verify(pk_B, IOU, \sigma_B) \neq true$ **then**
    $\zeta.Close()$
  if $IOU.I \neq \mathbb{B}$ *or* $IOU.OU \neq \mathbb{F}$ **then**
    $\zeta.Close()$
  if $IOU.price_{id_C} \neq price_{id_C}$ **then**
    $\zeta.Close()$
  $\sigma_F := Sign(sk_F, k)$
  $\zeta.Send(M_5 < k, \sigma_F >)$ to $\mathbb{B}$
  $\zeta.Store(M_4, M_5)$

Algorithm 7 Decrypt And Verify

function $\mathbb{B}$.DecryptAndVerify(ζ, k, σ_F)
  $c_1, \ldots, c_n = Dec^k(e_1), \ldots, Dec^k(e_n)$
  $id'_C = \{H(c_1), \ldots, H(c_n)\}$
  $< cid, reqid, id_C, price_{id_C} >= \zeta.Load(M_1).m$
  for each i *in* id_C **do**
    if $id_C[i] \neq id'_C[i]$ **then**
      ▷ Collect dispute evidence from channel
      $M = \zeta.Load(M_1, M_3, M_4, M_5)$
      $\zeta.Close()$
      $RaiseDispute(\mathbb{B}, \mathbb{F}, M, reqid, i, c_i)$
  $C =< c_1, \ldots, c_n >$
  Everything OK. Continue Trade

Algorithm 8 Dispute Resolve Smart Contract

$M =< M_1, M_3, M_4, M_5 >$ ▷ Dispute evidence

1: **function** RaiseDispute($\mathbb{B}$, $\mathbb{F}$, M, $reqid$, i, c_i)
2: $< id_E, \sigma_F, \sigma_B >\leftarrow M.M_3$
3: $< IOU, \sigma_B >, < k, \sigma_F >\leftarrow M.M_4,\ M.M_5$
4: **if** k is *null* or $H(E(k, c_i)) \neq id_E(i)$ **then**
5: Start timer τ; On Timeout:
6: **if** (No k from $\mathbb{F}$ within τ) OR
7: ($\mathbb{F}$ gives k **but** $H(E(k, c_i)) \neq id_E(i)$)
8: **goto** ReturnMoneyTo$\mathbb{B}$
9: $< cid, reqid, id_C, price_{id_C} >\leftarrow M.M_1.M_0$
10: **if** $H(c_i) = id_C(i)$ **then**
11: ▷ $\mathbb{B}$ Cheated, **No Loss to** $\mathbb{F}$
12: **return**
13: **else** ▷ $\mathbb{F}$ Cheated.
ReturnMoneyTo$\mathbb{B}$:
14: $\mathbb{F}.escrowBalance$ -= $price_{id_C}$
15: $\mathbb{B}.escrowBalance$ += $price_{id_C}$

Algorithm 9 Penalizer Smart Contract

bounty: penalty amount
cid: channel id for <$\mathbb{B}$ $\mathbb{F}$>

1: **function** $\mathbb{B}$.SubmitClaim($\mathbb{B}$, $\mathbb{F}$, cid, M_1, M_1')
2: $< cid, reqid, id_C, price_{id_C}, \sigma_B, \sigma_F >\leftarrow M_1$
3: $< cid', reqid', id_C', price'_{id_C}, \sigma_B', \sigma_F' >\leftarrow M_1'$
4: **if** $VerifySigns(\mathbb{B}, \mathbb{F}, [M_1, M_1']) \neq true$ **then**
5: ▷ Message signatures Invalid.
6: **return**
7: **if** (<$reqid$,cid> = <$reqid'$,cid'>) AND
8: ($id_C \neq id_C'$) **then**
9: ▷ $\mathbb{F}$-$\mathbb{B}$ collusion attack detected
10: ▷ Penalize $\mathbb{F}$ and reward $\mathbb{B}$
11: $\mathbb{F}.escrowBalance$ -= $bounty$
12: $\mathbb{B}.escrowBalance$ += $bounty$

As shown in the Fig. 2, in both phase 2 & 3, $\mathbb{B}$ & $\mathbb{F}$ **do not** interact with blockchain and carry out opportunistic exchanges (based on insight 2).

• **Phase 4: Settlement:** At the end of multiple exchanges, $\mathbb{F}$ and $\mathbb{B}$ submit all their offchain state to blockchain smart contracts (*trusted arbitrators*) to settle channel balance, resolve disputes and automatically pay $\mathbb{O}$ based on the number of successful exchanges.

Settle.1 Channel Closure and Settlement: Akin to state-channels, VADER channel closing semantics guarantee that once a party closes the channel, the other party has up to time τ to submit its set of offchain signed messages as evidence to the *channel settlement* smart contract. After time τ, the smart contract verifies the validity of each offchain message sequence and first settles the successful exchanges by transferring money worth $\sum_i price_{id_C}$ to $\mathbb{F}$ and $\sum_i (amt_O\%)$ to $\mathbb{O}$ from channel escrow. Disputes are settled as described next.

Settle.2 Dispute Resolution: In case of a dispute raised by $\mathbb{B}$ the *Dispute Resolve* Contract Alg. 8 performs two steps to identify the faulty participant as described below **1)** In line 4, it encrypts the disputed chunk with key k (if present) and checks if the hash of the encrypted chunk matches the one agreed by $\mathbb{B}$ in M_3. In case of *key not released* dispute by $\mathbb{B}$, $\mathbb{F}$ has time τ within which to submit k. **2)** In line 10, it computes hash of the chunk and checks if it matches with the hash registered by $\mathbb{O}$ in *Init.1*. A **faulty facilitator** will fail step (2) if chunk is indeed different from the one registered by $\mathbb{O}$; otherwise, the request is discarded because of **faulty buyer** behavior. Once the faulty party is detected, channel escrow balance is appropriately transferred to the honest party.

At the end of settlement and dispute resolution, offchain channel state is transferred on-chain, and application progresses, disputes continuing directly on blockchain with each party submitting messages directly to blockchain instead of each other. We note that VADER automatically pays out royalty to $\mathbb{O}$ based on the successful exchanges submitted to blockchain during settlement phase. Therefore, VADER guarantees $\mathbb{O}$ fairness as long as $\mathbb{B}$ and $\mathbb{F}$ *honestly* report their offchain messages on chain and do not collude with each other. We next describe how VADER handles collusion between a malicious $\mathbb{B}$ and $\mathbb{F}$ and still guarantees $\mathbb{O}$ fairness.

3.3 Preventing $\mathbb{F}$-$\mathbb{B}$ Collusion

The offchain execution of VADER between $\mathbb{B}$ and $\mathbb{F}$ makes $\mathbb{O}$ vulnerable to a collusion attack [2] where in $\mathbb{B}$ and $\mathbb{F}$ submit an altered set of offchain message exchanges during settlement to deny $\mathbb{O}$ of its fair share. We explain how such collusion can be undertaken next, followed by the description of the technique used to prevent such attacks.

Offchain Alternate Message Construction: Consider the case where $\mathbb{B}$ and $\mathbb{F}$ have completed a successful exchange of video id_C owned by $\mathbb{O}$. Note that at this point, $\mathbb{F}$ has already received *IOU* and

[2]The problem does not occur in case one of $\mathbb{B}$ or $\mathbb{F}$ are honest, as the adherence to protocol by any one party will force the other party to act honestly or loose out.

$\mathbb{B}$ the desired file. Therefore, both parties are incentivized to collude to maximize profit by constructing a new set of message exchanges simulating **1)** $\mathbb{B}$ requesting video id'_C owned by a sybil entity ($\mathbb{O}$') controlled by $\mathbb{F}$ benefitting $\mathbb{F}$, **2)** $\mathbb{F}$ agreeing for price $price_{id'_C}$ much lesser than $price_{id_C}$ benefiting $\mathbb{B}$. As part of alternate message construction, $\mathbb{B}$ creates a new Exchange Request Message M'_0, (m=<cid, $reqid$, $id_{C'}$, $price_{id'_C}$>,σ_B) with video id $id_{C'}$. Note that a *rational* $\mathbb{B}$ will reuse the <cid,$reqid$> pair, since creating either a new cid' or $reqid'$ makes $\mathbb{B}$ vulnerable to $\mathbb{F}$ submitting it as legitimate evidence of exchange to blockchain, and withdrawing $\mathbb{B}$s money. We leverage the above nuanced message construction to detect and penalize collusion.

Penalizer Smart Contract: Specifically, following insight 3, we introduce another smart contract, *Penalizer* Alg 9, which will pay $\mathbb{B}$ a large penalty from $\mathbb{F}$'s funds (deposited during channel funding in *Init.2*), if it can submit a pair of M_1, M'_1 messages, in line 2 & 3, having same *reqid* and *cid* but different content id's id_C, id'_C. The penalization scheme introduces an element of distrust between $\mathbb{B}$ and $\mathbb{F}$ and prevents collusion, forcing them to honestly report their offchain exchanges during settlement. In summary, our penalty scheme ensures that in a realistic setting, where most buyers are not controlled by $\mathbb{F}$, the $\mathbb{F}$ is disincentivized to collude with $\mathbb{B}$, for fear of paying a heavy penalty. Consequently our protocol guarantees fairness to all three parties viz. $\mathbb{O}$, $\mathbb{F}$ and $\mathbb{B}$.

Limitations We note that the offchain nature of VADER entails a buyer (or its delegatee [68]) to continuously remain *online*, and maintain *significant local state*, as well as lock up *sufficient liquidity* in channel till settlement. Consequently, VADER may not be appropriate for light weight clients that are ephemeral, lack local state or cannot afford to lock liquidity for long time, and depend solely on blockchain for tamper-proof logging and availability. We evaluate this model in Sec. 4.

Extension to multiple $\mathbb{O}$, $\mathbb{F}$, $\mathbb{B}$: In reality multiple parties can assume any of the roles in a single exchange (say multiple $\mathbb{O}$ of same content). Note that our protocols will work for even in such settings as long as all parties *do no completely trust each other*. This is due to the fact that any new entities participation will either be essential to the exchange and in which case, their concerns will be similar to that of an active party, whose rights are protected by VADER protocol. Alternatively, the party might not be essential for completing the exchange, in which case, their concerns will be similar to that of passive party which are also safeguarded by VADER protocol.

3.4 Security Analysis of VADER

In this section we show that VADER is secure against Royalty Manipulation, Content Mismatch and Content Stealing attacks (*Atk.1,2,3*) defined in Sec. 2. We reiterate that as mentioned in Sec. 2 we do not address the orthogonal issue of content piracy.

Theorem 3.1. *Under the assumptions that, a) the digital signature scheme is unforgeable, b) the cryptographic hash function is collision-resistant and c) under honest majority, blockchain attributes (smart-contract execution and ledger) are tamper-resistant, we show that VADER protects fairness of* ***honest and rational parties*** *against Atk.1, 2 & 3.*

• **Atk.1:** VADER ensures that in presence of at least honest $\mathbb{F}$ or $\mathbb{B}$, all exchanges are recorded onto the blockchain during **channel close** either successfully or as a dispute. In case both $\mathbb{F}$ & $\mathbb{B}$ are malicious, VADER introduces distrust to prevents collusion between them through incentive techniques mentioned in Sec.3.3, forcing them to record correct state on the blockchain. Further, in VADER $\mathbb{O}$'s royalty is calculated inside the blockchain trusted arbitrator (smart contract). So under the assumption (c) any royalty manipulation attack will not be feasible given honest majority of blockchain participants.

• **Atk.2 & 3:** We analyse *Atk.2, 3* in individual malicious parties and collusion cases as follows:

→ **Malicious $\mathbb{O}$:** A malicious $\mathbb{O}$ may target a content mismatch attack (*Atk.2*) by sending mismatching content (C, id_C) in *Init.1*, but for an honest $\mathbb{F}$ to accept mismatching content and hashes, $\mathbb{O}$ needs to find a collision in the hash function which is assumed impossible, making this attack infeasible. Content stealing (*Atk.3*) is useless for $\mathbb{O}$ as it owns the very content.

→ **Malicious $\mathbb{F}$:** For a malicious $\mathbb{F}$ to successfully launch a content mismatch (*Atk.2*), it has to send a content C' instead of C as requested by $\mathbb{B}$. An honest $\mathbb{B}$ will not accept that unless $\mathbb{F}$ manages to find a collision in H such that $H(C') = H(C)$. Hence, under collision-resistant hash function assumption, *Atk.2* does not occur. Content stealing (*Atk.3*) is meaningless for $\mathbb{F}$ and will not occur as it arguably owns the very content.

→ **Malicious $\mathbb{B}$:** A malicious $\mathbb{B}$ can try to mount content mismatch attack *Atk.2* by raising a fraudulent complaint against $\mathbb{F}$ after receiving the correct content. However, since $\mathbb{B}$ and $\mathbb{F}$ exchange a

co-signed message M_3 agreeing to $(E^k(C), id_E)$, $\mathbb{B}$ cannot generate $\mathbb{F}$s sign on a $(E^k(C'), id'_E)$ due to the unforgeability of signatures assumption. Similarly, it cannot raise a complaint with *Dispute Resolve* without submitting a mismatching chunk which when encrypted under k leads to the same hash, without finding a collision in the hash function, which is assumed impossible.

A malicious $\mathbb{B}$ cannot steal content(*Atk.3*) from $\mathbb{F}$ as as an honest $\mathbb{F}$ will not transfer the decryption key k without payment in message M_4 and *Dispute Resolve* contract ensures that $\mathbb{B}$ provides *IOU* to $\mathbb{F}$ before handing over the key k.

→ $\mathbb{O}$-$\mathbb{F}$ **Collusion:** A malicious $\mathbb{O}$ cannot help $\mathbb{F}$ with content mismatch attack (*Atk.2*) to cheat $\mathbb{B}$ by providing wrong content as an honest $\mathbb{B}$ requests for a particular content in the Xchg.1 step. Since $\mathbb{O}$ has no control over this step, this attack reduces to a malicious $\mathbb{F}$ case, which we already explained the protocol to be secure against. Also both $\mathbb{O}$ and $\mathbb{F}$ arguably own the content, hence obviating the need for content stealing attack (*Atk.3*).

→ $\mathbb{B}$-$\mathbb{O}$ **Collusion:** Content mismatch (*Atk.2*) & stealing (*Atk.3*) attacks in this case would be $\mathbb{B}$ and $\mathbb{O}$ trying to exchange content through VADER but denying payment to $\mathbb{F}$. To this end, $\mathbb{O}$ may try to submit wrong (C,id_C) in Init.1 step, in order for $\mathbb{B}$ to raise a dispute. An honest $\mathbb{F}$ will not countersign a mismatching (C, id_C), unless $\mathbb{O}$ finds a collision in H which by assumption is impossible. Apart from Init.1, $\mathbb{O}$ does not participate in the remaining protocol, reducing these attacks to the malicious $\mathbb{B}$ case, which the protocol is secure against.

Note that the case of mutually known $\mathbb{O}$ and $\mathbb{B}$ carrying out entire exchange among themselves without involving $\mathbb{F}$, is orthogonal to the VADER setting wherein $\mathbb{B}$ and $\mathbb{O}$ do not know each other.

→ $\mathbb{B}$-$\mathbb{F}$ **Collusion:** Rational $\mathbb{B}$ and $\mathbb{F}$ will not let a content mismatch attack (*Atk.2*) and content stealing attack (*Atk.3*) to occur respectively as they are hurt by the same. We ignore the case where a $\mathbb{F}$ (or any fake $\mathbb{O}$') sells the content to a $\mathbb{B}$ outside our ecosystem as that is a case of content piracy.

Under all the possible cases (corruption and collusion), VADER is secure against *Atk.1, 2 and 3*, thus completing the proof. We briefly highlight the reason for VADER to be secure even with multiple $\mathbb{F}$, $\mathbb{O}$ or $\mathbb{B}$ in Sec. 3.3.

4 IMPLEMENTATION

We compare the performance of VADER against the following two baselines

VANILLA: We implemented VANILLA to emulate traditional HTTPS based video delivery as described in RFC [72]. VANILLA protocol **does not** use blockchain enabling us to benchmark performance overhead of VADER over state-of-art mechanisms.

Blockchain Mediated Exchange (BME): To understand the benefits of batching protocol messages, we implement BME protocol wherein $\mathbb{F}$ and $\mathbb{B}$ progress application state directly on blockchain instead of directly sending to each other as in VADER. In BME, akin to VADER, video is transferred offchain in encrypted format, while each exchange is directly settled onchain before starting the next. We note that in BME parties need not open & close a channel, and some messages can be batched while committing to blockchain, requiring only 3 commits (i.e. <M_0,M_1>, <M_2,M_3,M_4> & <M_5>) in *happens before order*. We incorporate the above optimizations in our implementation of BME.

Application Prototype: We implement $\mathbb{F}$ as a Django webapp (868 SLOC), $\mathbb{O}$ (256 SLOC) & $\mathbb{B}$ apps for VADER (699 SLOC), VANILLA (348 SLOC) and BME (617 SLOC), as Python applications.

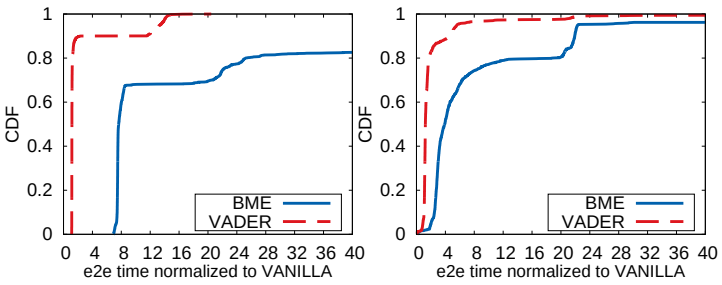
Smart Contracts: We implement all VADER and BME functionalities as chaincodes (Smart Contracts) on top of Hyperledger Fabric [52] v1.2 (Fabric) [23]. We implement chaincode utilities for time estimation based on block height and conditional (time and condition locked) escrow accounts (no native crypto-currency in Fabric v1.2).

Maliciousness: We emulate a malicious $\mathbb{F}$ as one that sends non-matching chunk hashes to $\mathbb{B}$. Similarly, a malicious $\mathbb{B}$ is emulated by raising a dispute with the *Dispute Resolve* smart contract even after receiving the correct video from $\mathbb{F}$.

5 EVALUATION

In our evaluation, we answer the following, **1)** the performance overhead of VADER compared to baseline protocols, **2)** the amortization benefits of VADER, **3)** the effect of maliciousness on VADER performance, **4)** the sensitivity of VADER to the underlying blockchain platform.

• **Experimental Setup:** We run our experiments on 91 VMs (Ubuntu, 2.1GHz 16 CPU, 32GB, SSD) in Softlayer Cloud [7], across five geo-distributed data centers spanning four continents. Based on benchmarking, average latency was estimated at 0.4ms (and throughput 5Gbps) for intra-DC and varied 21.8-337ms (throughput 460-35.6Mbps) for inter-DC network. All experiments were conducted with a Fabric network consisting of 10 blockchain peers (one per $\mathbb{F}$ reflecting a network run by the



(a) CDN Topology (b) Random Topology

Figure 4: E2E time of VADER and BME normalized to VANILLA with #𝔹 500, #𝔽 10, 10% maliciousness exchanging random #files 10-250

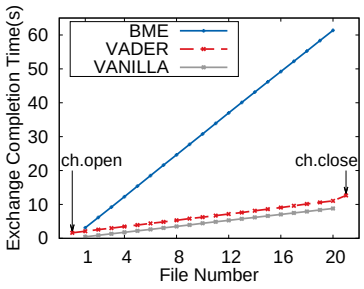


Figure 5: File exchange timeline for an honest 𝔹 exchanging 20 files, 20MB each under CDN topology

𝔽s) running the default ordering service with out of the box configuration parameters [1].

• **Performance Metrics:** We use the following metrics for quantifying performance overhead of the three schemes:**1) end-to-end (e2e) time** measures time elapsed from the instant 𝔹 requests a file till it's last chunk is ready for consumption (downloaded and decrypted) is our primary metric; **2) component contributions** measured as the fraction of time spent in each sub component obtained by dividing e2e time into three components– **a) Protocol Time** is the total time spent in blockchain interactions and message exchanges between the parties; **b) Transfer Time** is the time taken for transferring encrypted chunks; **c) Verify Time** is the time spent by 𝔹 in decrypting and verifying hash of the exchanged file. In the presence of maliciousness, this sub-component includes the time spent in interacting with the dispute resolution smart-contract;

• **Experimental Methodology:** We benchmark the performance of VANILLA, VADER & BME under different realistic conditions by varying the underlying network topology, load on 𝔽 and overall maliciousness in the system. Unless stated otherwise, we run our experiments for five iterations and report average results.

5.1 Macro Benchmarks

In this section we benchmark the performance overhead of all three schemes under realistic settings. We run 500 𝔹s in 5 DCs (100 𝔹s per DC, equally load-balanced among 5 VMs). The 𝔹s are configured to exchange a 20MB file in chunks of 512KB [3], random number of times (10 to 250, in increments of 5). We run 10 𝔽s in 10 VMs equally distributed across the 5 DCs. We configure 10% of the 𝔽s to be malicious (as described in Sec. 4).

***5.1.1* CDN topology:** We emulate a CDN like hierarchical topology implemented by real world facilitators [20, 87] where a 𝔹 exchanges content from the nearest (same DC) 𝔽 and measure e2e time for all the three schemes.

Performance Overhead: Non-Malicious: In Fig. 4a, we plot the CDF of (median) e2e time for VADER and BME 𝔹s, normalized to VANILLA. VADER adds only minimal overhead to VANILLA ranging from min. 12% to 16% at the median, making it a practically deployable system for providing fairness at scale. Notably 80% of the VADER 𝔹s have an overhead of less than 23%. In contrast, BME has an overhead ranging from min. 690% to 764% at the median. This is due to the fact that VADER interacts with blockchain only twice (channel open and close) over multiple successful exchanges, while BME interacts with blockchain thrice for each exchange.

In summary, VADER is able to amortize blockchain interaction time over multiple exchanges leading to minimal performance overhead compared to VANILLA (16%) whereas, lack of amortization leads to significant degradation in BME (764%).

Performance overhead: Malicious: We also observe that maliciousness (10% of 𝔽s) causes severe performance degradation shown by a lot worse performance (notch at 90%) for both VADER and BME. VADER adds an overhead of atleast 1160% while BME adds 9400% respectively for clients of malicious 𝔽s. The performance degradation can be attributed to the overhead imposed by onchain execution of *Dispute Resolve* contract involving sign verification and waiting for settlement timeouts. as described in Alg. 8.

[3]Empirically determined to be optimal, results omitted for sake of brevity.

Blockchain	Bitcoin	Ethereum	Litecoin	Siacoin	Monero	Zcash	Peercoin	Dogecoin
Consensus	PoW	PoW	PoW	PoW	PoW	PoW	PoW & PoS	Pow
Block Gen. Time(s)	545.52	14.58	149.82	600.00	121.56	150.00	484.38	62.52
VADER (s)	5.88	0.57	1.92	6.42	1.64	1.92	5.26	1.05
BME (s)	1637.04	44.22	449.95	1800.48	365.17	450.48	1453.63	188.04

Table 1: VADER & BME average e2e expected latencies (per 20MB file) for public blockchain networks while batching 200, 20MB file exchanges in a session. Block generation time for Siacoin is from [4], for rest, 1/1/2019 from [3]. PoW = Proof Of Work, PoS = Proof of Stake

5.1.2 **Random topology:** Next we evaluate the performance of VADER under a different network topology viz. random. We repeat the same experiment above, but allowing $\mathbb{B}$s to exchange content from a random $\mathbb{F}$ in any DC this time [4].

Fig. 4b shows that compared to VANILLA, VADER has a minimal overhead of 23% at the median compared to BME's 391%. Interestingly, BME has lesser overhead in this scenario compared to the CDN scenario. This is due to the inter-dc network conditions that make network transfer times relatively worse for a large number of $\mathbb{B}$s across all three schemes. Consequently, overall network transfer time increases while the blockchain overhead remains the same making BME incur relatively lesser overhead compared to CDN topology.

Note that, VADER maintains its minimal overhead (23%) compared to VANILLA even under adverse network conditions making it deployable over a variety of network topologies.

5.1.3 **Analysis of a single CDN experiment:** To get a better understanding of when a file is available for viewing by a $\mathbb{B}$ (startup latency), we randomly select a single $\mathbb{B}$ from the CDN experiment. and depict the timeline of all file exchanges (20MB, 20 times), for all three schemes in Fig. 5. We note that, barring the one time channel open delay (1.61s), VADER adds only a minimal delay compared to VANILLA for each file, thereby not adversely affecting user experience. On the other hand, BME adds significant delay per file due to its three commit blockchain overhead added to each file. *We also observe that, barring channel open and close overheads (3.2s), VADER adds only 7% delay over vanilla, making it a viable system.*

5.1.4 **Estimated performance on Public Blockchain Networks:** We evaluate the sensitivity of VADER to underlying blockchain platform by estimating it's performance on various public blockchains listed in Tab. 1. For this study, we calculate the median e2e time of a single file for VADER & BME and isolate it into two components viz. 'blockchain protocol' time and 'miscellaneous' time involving network transfer and crypto operations. We estimate blockchain protocol time for VADER by dividing twice the block generation time of the underlying blockchain (one each for channel open-close) by the number of file exchanges. For BME we calculate protocol time as thrice the block generation time (corresponding to the three blockchain commits). Finally we add the miscellaneous overhead for both protocols to get the projected time taken by each.

We observe that even in the case of public blockchains, VADER's amortization benefits drastically outperform BME. In the case of blockchain networks with high block generation time (such as Bitcoin) VADER is able to achieve 27.21Mbps, making it practical even on public blockchains, while BME throttles down to 0.01Mbps. *On the other hand, in the case of a blockchain like Ethererum with lower block generation time, VADER can achieve nearly 280Mbps, which is comparable to VANILLA's 384Mbps.*

5.2 Micro benchmarks

In this section, we validate our design choices by benchmarking the performance of various system sub components. We allocate 10 VMs running a single $\mathbb{F}$ each, and run 200 $\mathbb{B}$s in 20VMs (20 buyers per vm). We configure $\mathbb{B}$s to exchange a 20MB file in chunks of 512KB, 50 times randomly from any $\mathbb{F}$. Default maliciousness is set to 0% (unless specified otherwise).

5.2.1 **e2e time vs #files:** Fig. 6 shows the effect of increasing #files (5 to 200) on average e2e time per file for a constant file size of 20MB. We observe that VADER overhead compared to VANILLA keeps decreasing with increasing no. of files, from 90%(5 files) to 13%(200 files). Clearly, VADER is able to

[4] Due to inter-dc network limitations and also to reduce the financial cost of the experiment, we restrict our file size to 5MB, keeping number of exchanges same (10 to 250). (To validate results, we have manually run experiments with 20MB files and found similar trends as reported).

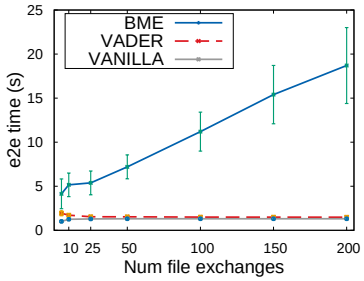


Figure 6: E2E time amortization w. increasing no. of file exchanges. (filesize 20MB)

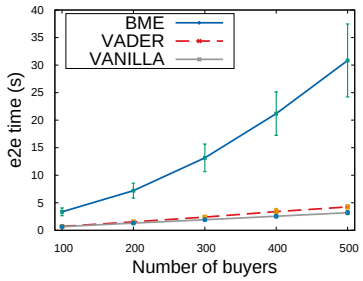


Figure 7: E2E time vs increasing #𝔹s (20MB, 50 times)

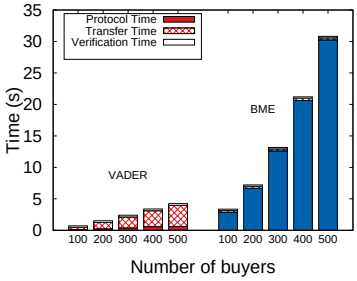


Figure 8: Per component e2e time contribution vs #𝔹s (20MB, 50 times)

amortize blockchain overhead over multiple files. On the other hand, note that the e2e time for BME increases rapidly with the no. of files, from 4.14s(5) to 18.7s(200) since BME bottlenecks by hitting the blockchain for each file.

***5.2.2* e2e time vs #buyers:** Next we study the effect of loading 𝔽 by increasing no. of 𝔹s from 10 to 50 per 𝔽, keeping filesize (20MB) and #files (50) constant. In Fig. 7, we observe that VADER's overhead increases from a mere 7% (0.71s) at 100 𝔹s to 31% (4.24s) at 500 𝔹s. This is explained by the extra load imposed on the 𝔽s by the signing and verification, steps of VADER. In contrast, BME's e2e time increases rapidly from 3.35s to 30.84s, due to the extra load on the blockchain layer (replicated execute and consensus) imposed by the increasing no. of commits, from increasing 𝔹 count.

***5.2.3* Component Analysis:** Fig. 8 shows the component contribution (defined in Metrics, Sec. 5) towards e2e time of a file for VADER and BME protocols with increasing #𝔹s per 𝔽. We observe that for VADER there is a 10x increase in network transfer time (from 0.34s to 3.35s), while protocol time increases 5x from 0.125s to 0.61s with increasing 𝔹s. The large network transfer time increase in VADER is attributed to the fact that an increasing no. of 𝔹s simultaneously fetch content (flash crowd) thereby putting more stress on 𝔽s and the underlying network. (We note that VANILLA network transfer time behaves in a similar fashion). On the other hand, the protocol time for BME increases rapidly (from 2.84s to 29.7s) with increasing #𝔹s while the network transfer time remains the same at around 0.34s. This is due to the fact that in the case of BME 𝔹s, the blockchain commit wait times provide a more spaced out execution and 𝔽 network time remains constant.

***5.2.4* Effect of Maliciousness:** We study the effect of maliciousness by increasing maliciousness from 0% to 100% and plot average e2e time for VADER and BME in Fig. 9. [5] We observe that VADER starts with lower e2e time (1.53s) compared to BME (7.2s) for 0% maliciousness, while VADER e2e time increases with increasing maliciousness and coming closer to BME. This is due to the fact that after the first dispute, VADER closes channel and continues application progress directly on blockchain similar to BME.

In the worst case of 100% maliciousness, VADER begins to have higher completion time compared to BME due to the additional channel open-close overhead. In Fig. 10, it is interesting to note that in case of VADER, even in the presence of maliciousness, the non-malicious parties remain unaffected, and only 𝔹s interacting with a malicious 𝔽 incur higher e2e time. For example, e2e time for 60% of 𝔹s is less than 1.65s with 40% maliciousness and jumps to 5.76s right after; and it is less than 1.59s for 20% of 𝔹s in 80% maliciousness, after which it jumps to 6.34s.

6 RELATED WORK

In this section, we review and contrast our work with existing work on building fair, auditable and decentralized systems.

• **Blockchain Platforms:** Beginning with Bitcoin [70], the recent past has seen the emergence of a number of blockchain platforms [33, 44, 52, 76, 77, 85], that provide a tamper proof immutable ledger of transactions as well as smart contract capabilities secured by an underlying consensus protocol. The various blockchain platforms vary in their choice

[5] VANILLA does not have a dispute resolution mechanism, hence, we don't run VANILLA in experiments benchmarking maliciousness.

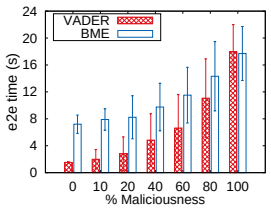
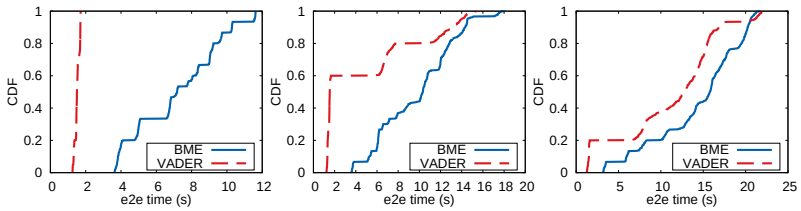


Figure 9: System wide avg e2e time per file vs % maliciousness (20MB, 50 times)



(a) 0% Malicious **(b) 40% Malicious** **(c) 80% Malicious**

Figure 10: CDF(s) of e2e time per $\mathbb{B}$ under increasing maliciousness. (20MB, 50 times)

of consensus protocol, transaction privacy models, as well as membership in the blockchain network. Permissioned/consortium model restricts blockchain access to only a few users/organizations, while permissionless model allows any node to join and participate in the network. Typical permissioned blockchain platforms also support *access control* capabilities at the smart contract layer to provide confidentiality guarantees to higher layer blockchain applications. *VADER relies on granular access control, as well as tamper-proof smart contract execution and auditability of blockchain platforms, to guarantee multi-party fair exchange, even in the presence of passive participants.*

• **Blockchain Scalability:** The underlying consensus protocol overhead limits the scalability of blockchains. Recent solutions to addressing scalability broadly fall in two buckets.

Layer 1 Solutions: These solutions involve making *onchain* modifications to the underlying blockchain design such as changing the block size [31] or block generation time [63]. The authors in [43] adopt a leader election based consensus algorithm for achieving faster consensus and block generation fairness in bitcoin. [22, 38, 58, 65] implement sharding to scale transaction throughput wherein the main blockchain is divided into multiple independent shards within which transactions can be validated in parallel, and finally merged into the main chain. Algorand [46] uses a committee based consensus protocol, while RapidChain [91] uses a mix of sharding and committee based consensus to improve the scaling further. *We note that even if the above works are able to improve transaction throughput to a few thousand transactions per second, blockchain consensus will still impose a ceiling on transaction throughput*

Layer 2 Solutions: An alternate design to scale throughput is to decouple transaction processing speed from the underlying blockchain protocol by reducing the number of interactions with the blockchain. Payment channels [74, 79] and generalized state channels [37, 42, 69, 73, 82, 84] enable parties to directly transact with each other off-chain and finally batch multiple transactions into a single blockchain transaction. This allows applications to scale independent of the blockchain protocol while offering security guarantees at par with native blockchain. While state channels help in scaling, they require **1)** all parties to be known to each other apriori and **2)** all parties (or their delegatee [68]) to continuously remain online. *On the other hand, VADER works even when parties are not known to each other apriori and one of the parties is passive and offline (content owner).*

• **Fair Exchange:** Fair exchange is a well studied problem, especially fairness for electronic commerce [25]. Two party fair exchange without a trusted third party (TTP) is known to be impossible [47, 71, 78, 89] without relaxing security requirements. Recent works have looked at designing blockchain based protocols for fair exchange [24, 56, 60, 61]. Bentov et.al [29] describe a bitcoin based *claim-or-refund* framework for fair exchange in which either parties receive the intended goods or are compensated monetarily. Building on this, the authors in FairSwap [41] describe a framework that enable a sender and receiver to exchange digital goods such as files in a fair manner through the use of a 'judge' smart contract. Choudhuri et.al[36] design protocols that ensure fairness in secure multi-party computation using blockchain-like primitives. *The above protocols guarantee fairness only between the active parties and do not cover passive participants, whereas VADER guarantees fairness even for passive parties.*

• **Decentralized Marketplaces:** Recent works on blockchain based decentralized marketplaces allow buyers and sellers to trade directly with each other without a centralized platform [27, 55, 57, 62, 83]. In these systems, dispute resolution is handled offline in an ad-hoc manner, providing no guarantees on fairness. *VADER on the other hand guarantees fair exchange and is better aligned with*

existing centralized marketplaces.

• **Video Streaming:** Most commercial video streaming services [14, 16] follow RFC [72] that describes the streaming of video over HTTP(S). There has been significant research in the design, measurement and characterization and optimization of scalable video streaming systems like [20, 21, 28, 39, 45, 59, 75, 87] characterize the end to end performance of a commercial video streaming service and identify a number of bottlenecks across different layers of the stack. Similarly, [87] studies policies used for server selection in the Youtube CDN network. while the authors in [26] study the effects of CDN augmentation techniques such as P2P-CDN and telco-CDN federation on video workloads. Similarly, on the client side, a number of bitrate adaptation algorithms have been developed with the goals of minimizing buffering, start up latency, improving video smoothness etc. *We note that above prior work's focus on improving end user experience ard orthogonal to VADER's focus on guaranteeing fairness.*

• **Auditing Mechanisms:** A number of works have looked at auditing running systems. PeerReview [49] leverages tamper-evident logging to detect when a node deviates from the expected behaviour. AVMs [48] on the other hand, use logging to record all incoming/outgoing messages from a VM and ensure correct execution of a remote system. [30] helps ad system operators debug revenue problems through multi-dimensional analysis of various metrics. However, the approach requires access to logs and other internal system metrics. Such works are not applicable in our setting as we do not trust facilitators to act honestly. Recent works have also focussed on auditing the working of web systems [34, 35, 50, 51, 53, 54] through black box measurements to detect violations in application defined fairness. *However, these works are limited to detecting unfair practices and do not provide any mechanisms for guaranteeing fairness of the participants.*

• **Complementary Technologies:** We highlight complementary technologies that VADER can leverage to further enhance the security of the platform. Mangipudi et.al [66] use a combination of content watermarking and on-chain penalty mechanisms to prevent content piracy which can be easily embedded into VADER smart contracts. Emerging technologies such as Intel SGX [8] that provide computational integrity and verifiability through hardware mechanisms could be leveraged for protecting the confidentiality of videos stored on the video server.

7 CONCLUSION

We introduce the problem of *Multi-party fair exchange* for digital assets, which requires safeguarding the rights of active and passive participants. We propose a protocol to ensure *Multi-party fair exchange* for digital assets, by leveraging blockchain and intelligent incentive alignment. We build a prototype of the protocol on Hyperledger Fabric and extensively evaluate performance of our approach across a realistic test bed and show results that demonstrate the feasibility of our system.

REFERENCES

[1] 2018. Hyperledger Fabric Github. https://github.com/hyperledger/fabric/tree/release-1.2

[2] 2019. Apple introduces 14-day return on iTunes, scaring coders and musicians. Online; https://www.theguardian.com/. https://bit.ly/2INFCQv

[3] 2019. Bitinfocharts Block Generation Time. "Online; https://bitinfocharts.com". https://bit.ly/31SlDYi

[4] 2019. Coingecko Siacoin Details. https://www.coingecko.com/en/coins/siacoin

[5] 2019. Etsy - Digital Prints. https://www.etsy.com/in-en/market/digital_prints

[6] 2019. Google play store. https://play.google.com/store

[7] 2019. IBM Softlayer Cloud Docs. "Online; IBM Cloud Docs". https://ibm.co/2J9PXVx

[8] 2019. Intel Software Guard Extensions (SGX). https://software.intel.com/en-us/sgx

[9] 2019. New Scam Holds YouTube Channels for Ransom. "Online; https://www.bleepingcomputer.com". https://bit.ly/2XxqT4e

[10] 2019. Not getting paid at all? https://support.google.com/youtube/thread/4347574.

[11] 2019. Spotify. https://www.spotify.com

[12] 2019. Steam. https://store.steampowered.com/

[13] 2019. The global video streaming market size. "Online; https://www.reportbuyer.com/". https://www.reportbuyer.com/product/5763843/?

[14] 2019. Vimeo. https://vimeo.com/

[15] 2019. WideVine. https://www.widevine.com/.

[16] 2019. Youtube. https://www.youtube.com/

[17] 2019. YouTube CEO addresses top creator issues including copyright claims and trending section. "Online; https://www.theverge.com/". https://bit.ly/2II3Knt

[18] 2019. YouTube creators have complained about declines in ad revenue. "Online; https://www.vox.com/". https://bit.ly/2ZU6TGH

[19] 2019. YouTube's small creators pay price of policy changes after Logan Paul scandal. "Online; https://www.theguardian.com/". https://bit.ly/2DmeC5c

[20] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. 2012. Unreeling netflix: Understanding and improving multi-CDN movie delivery. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*. 1620–1628. https://doi.org/10.1109/INFCOM.2012.6195531

[21] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. 2012. Vivisecting YouTube: An active measurement study. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*. 2521–2525.

https://doi.org/10.1109/INFCOM.2012.6195644

[22] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. 2018. Chainspace: A Sharded Smart Contracts Platform. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*.

[23] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*. 30:1–30:15. https://doi.org/10.1145/3190508.3190538

[24] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2016. Secure multiparty computations on Bitcoin. *Commun. ACM* 59, 4 (2016), 76–84. https://doi.org/10.1145/2896386

[25] N Asokan. 1998. *Fairness in Electronic Commerce*. PhD Thesis. University of Waterloo, Canada. https://asokan.org/asokan/research/Asokan98.pdf

[26] Athula Balachandran, Vyas Sekar, Aditya Akella, and Srinivasan Seshan. 2013. Analyzing the Potential Benefits of CDN Augmentation Strategies for Internet Video Workloads. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC '13)*. ACM, New York, NY, USA, 43–56. https://doi.org/10.1145/2504730.2504743

[27] Open Bazaar. 2019. OpenBazaar: Online Marketplace | Peer-to-Peer Ecommerce. https://www.openbazaar.org/

[28] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. 2019. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Communications Surveys and Tutorials* 21, 1 (2019), 562–585. https://doi.org/10.1109/COMST.2018.2862938

[29] Iddo Bentov and Ranjit Kumaresan. 2014. How to Use Bitcoin to Design Fair Protocols. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*. 421–439. https://doi.org/10.1007/978-3-662-44381-1_24

[30] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. 2014. Adtributor: Revenue Debugging in Advertising Systems. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014*. 43–55.

[31] Bitcoincash. 2019. BitCoin Cash. https://www.bitcoincash.org/

[32] BitTube. 2019. BitTube. https://bit.tube/

[33] Zero cash. 2019. Zero Cash. https://www.zerocash-project.org/

[34] Le Chen, Alan Mislove, and Christo Wilson. 2015. Peeking Beneath the Hood of Uber. In *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*. 495–508. https://doi.org/10.1145/2815675.2815681

[35] Le Chen, Alan Mislove, and Christo Wilson. 2016. An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*. 1339–1349. https://doi.org/10.1145/2872427.2883089

[36] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. 2017. Fairness in an Unfair World: Fair Multiparty Computation from Public Bulletin Boards. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. 719–728. https://doi.org/10.1145/3133956.3134092

[37] Jeff Coleman, Liam Horne, and Li Xuanji. 2019. Counterfactual: Generalized State Channels. https://l4.ventures/papers/statechannels.pdf

[38] George Danezis and Sarah Meiklejohn. 2016. Centrally Banked Cryptocurrencies. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*.

[39] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the Impact of Video Quality on User Engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*. ACM, New York, NY, USA, 362–373. https://doi.org/10.1145/2018436.2018478

[40] DTube. 2019. DTube. https://d.tube/

[41] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. 2018. FairSwap: How To Fairly Exchange Digital Goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 967–984. https://doi.org/10.1145/3243734.3243857

[42] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. 2018. General State Channel Networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 949–966. https://doi.org/10.1145/3243734.3243856

[43] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. 2016. Bitcoin-NG: A Scalable Blockchain Protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*. 45–59.

[44] Ethereum Foundation. 2019. Ethereum. https://www.ethereum.org/

[45] Mojgan Ghasemi, Partha Kanuparthy, Ahmed Mansy, Theophilus Benson, and Jennifer Rexford. 2016. Performance Characterization of a Commercial Video Streaming Service. In *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*. ACM, New York, NY, USA, 499–511. https://doi.org/10.1145/2987443.2987481

[46] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. 51–68. https://doi.org/10.1145/3132747.3132757

[47] O. Goldreich, S. Micali, and A. Wigderson. 1987. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*. ACM, New York, NY, USA, 218–229. https://doi.org/10.1145/28395.28420

[48] Andreas Haeberlen, Paarijaat Aditya, Rodrigo Rodrigues, and Peter Druschel. 2010. Accountable Virtual Machines. In *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, October 4-6, 2010, Vancouver, BC, Canada, Proceedings*. 119–134.

[49] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. 2007. PeerReview: practical accountability for distributed systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*. 175–188. https://doi.org/10.1145/1294261.1294279

[50] Aniko Hannak, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. 2014. Measuring Price Discrimination and Steering on E-commerce Web Sites. In *Proceedings of the 2014 Internet Measurement Conference, IMC 2014, Vancouver, BC, Canada, November 5-7, 2014*. 305–318. https://doi.org/10.1145/2663716.2663744

[51] Aniko Hannak, Claudia Wagner, David García, Alan Mislove, Markus Strohmaier, and Christo Wilson. 2017. Bias in Online Freelance Marketplaces: Evidence from TaskRabbit and Fiverr. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW 2017, Portland, OR, USA, February 25 - March 1, 2017*. 1914–1933. http://dl.acm.org/citation.cfm?id=2998327

[52] Hyperledger. 2019. Hyperledger Fabric. https://www.hyperledger.org/projects/fabric

[53] Costas Iordanou, Claudio Soriente, Michael Sirivianos, and Nikolaos Laoutaris. 2017. Who is Fiddling with Prices?: Building and Deploying a Watchdog Service for E-commerce. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*. 376–389. https://doi.org/10.1145/3098822.3098850

[54] Shan Jiang, Le Chen, Alan Mislove, and Christo Wilson. 2018. On Ridesharing Competition and Accessibility: Evidence from Uber, Lyft, and Taxi. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 863–872. https://doi.org/10.1145/3178876.3186134

[55] Oliver R. Kabi and Virginia N. L. Franqueira. 2018. Blockchain-Based Distributed Marketplace. In *Business Information Systems Workshops - BIS 2018 International Workshops, Berlin, Germany, July 18-20, 2018, Revised Papers*. 197–210. https://doi.org/10.1007/978-3-030-04849-5_17

[56] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. 2016. Fair and Robust Multi-party Computation Using a Global Transaction Ledger. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*. Springer-Verlag New York, Inc., New York, NY, USA, 705–734. https://doi.org/10.1007/978-3-662-49896-5_25

[57] Markus Klems, Jacob Eberhardt, Stefan Tai, Steffen Härtlein, Simon Buchholz, and Ahmed Tidjani. 2017. Trustless Intermediation in Blockchain-Based Decentralized Service Marketplaces. In *Service-Oriented Computing - 15th International Conference, ICSOC 2017, Malaga, Spain, November 13-16, 2017, Proceedings*. 731–739. https://doi.org/10.1007/978-3-319-69035-3_53

[58] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. 583–598. https://doi.org/10.1109/SP.2018.000-5

[59] Dilip Kumar Krishnappa, Samamon Khemmarat, Lixin Gao, and Michael Zink. 2011. On the Feasibility of Prefetching and Caching for Online TV Services: A Measurement Study on Hulu. In *Passive and Active Measurement - 12th International Conference, PAM 2011, Atlanta, GA, USA, March 20-22, 2011. Proceedings*. 72–80. https://doi.org/10.1007/978-3-642-19260-9_8

[60] Ranjit Kumaresan and Iddo Bentov. 2016. Amortizing Secure Computation with Penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, USA, 418–429. https://doi.org/10.1145/2976749.2978424

[61] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. 2016. Improvements to Secure Computation with Penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, USA, 406–417. https://doi.org/10.1145/2976749.2978421

[62] LBRY. 2019. LBRY - Content Freedom. https://lbry.io/

[63] Litecoin. 2019. Litecoin. https://litecoin.org/

[64] Livepeer. 2019. Livepeer - Peer to peer video services. Incentivized. https://livepeer.org/

[65] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. 17–30. https://doi.org/10.1145/2976749.2978389

[66] Easwar Vivek Mangipudi, Krutarth Rao, Jeremy Clark, and Aniket Kate. 2018. Automated Penalization of Data Breaches using Crypto-augmented Smart Contracts. *IACR Cryptology ePrint Archive* 2018 (2018), 1050. https://eprint.iacr.org/2018/1050

[67] Lee Marshall. 2015. âĂŸLet's keep music special. FâĂŤSpotifyâĂŹ: on-demand streaming and the controversy over artist royalties. *Creative Industries Journal* 8, 2 (2015), 177–189. https://doi.org/10.1080/17510694.2015.1096618

[68] Patrick McCorry, Surya Bakshi, Iddo Bentov, Andrew Miller, and Sarah Meiklejohn. 2018. Pisa: Arbitration Outsourcing for State Channels. *IACR Cryptology ePrint Archive* 2018 (2018), 582. https://eprint.iacr.org/2018/582

[69] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. 2017. Sprites: Payment Channels that Go Faster than Lightning. *CoRR* abs/1702.05812 (2017). arXiv:1702.05812 http://arxiv.org/abs/1702.05812

[70] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system,âĂİ http://bitcoin.org/bitcoin.pdf.

[71] Henning Pagnia and Felix C. GÃďrtner. 1999. *On the impossibility of fair exchange without a trusted third party*. Technical Report.

[72] Roger Pantos and William May. 2017. HTTP Live Streaming. *RFC* 8216 (2017), 1–60. https://doi.org/10.17487/RFC8216

[73] Perun. 2019. Perun Network. https://www.perun.network/

[74] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf

[75] Yanyuan Qin, Shuai Hao, Krishna R. Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue.

2018. ABR streaming of VBR-encoded videos: characterization, challenges, and solutions. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2018, Heraklion, Greece, December 04-07, 2018*. 366–378. https://doi.org/10.1145/3281411.3281439

[76] JPMC Quorum. 2019. Quorum - A permissioned implementation of Ethereum supporting data privacy. https://github.com/jpmorganchase/quorum

[77] R3. 2019. Corda. https://www.corda.net/

[78] Michael O. Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptology ePrint Archive* 2005 (2005), 187. http://eprint.iacr.org/2005/187

[79] Raiden. 2019. Raiden Network. https://raiden.network/

[80] Sandvine. 2018. "The Global Internet Phenomena Report October 2018". Online; https://www.sandvine.com/. https://bit.ly/2zNZBde

[81] Henri Sivonen. 2013. "Encrypted Media Extension: EME-DRM". https://hsivonen.fi/eme/

[82] Spankchain. 2019. Spankchain A blockchain based payment processor and live video streaming platform. http://spankchain.com

[83] Hemang Subramanian. 2018. Decentralized blockchain-based electronic marketplaces. *Commun. ACM* 61, 1 (2018), 78–84. https://doi.org/10.1145/3158333

[84] FunFair Technologies. 2019. Funfair Blockchain solutions for gaming. https://funfair.io/

[85] Tendermint. 2019. Tendermint. https://tendermint.com/

[86] Additional Changes to the YouTube Partner Program (YPP) to Better Protect Creators. 2018. How to earn money on YouTube. Online; https://youtube-creators.googleblog.com. https://bit.ly/2raNyUQ

[87] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M. Munafo, and Sanjay Rao. 2011. Dissecting Video Server Selection Strategies in the YouTube CDN. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems (ICDCS '11)*. IEEE Computer Society, Washington, DC, USA, 248–257. https://doi.org/10.1109/ICDCS.2011.43

[88] Vivuly. 2019. Vivuly - Decentralized Video Sharing. http://viuly.com/

[89] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS '86)*. IEEE Computer Society, Washington, DC, USA, 162–167. https://doi.org/10.1109/SFCS.1986.25

[90] YouTube. 2019. YouTube for Press. https://www.youtube.com/yt/about/press/

[91] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling Blockchain via Full Sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 931–948. https://doi.org/10.1145/3243734.3243853