Blockchain-based Invoice Factoring: from business requirements to commitments*

Ettore Battaiola[1], Fabio Massacci[2], Chan Nam Ngo[2], and Pierantonia Sterlini[2]

[1] Cassa Centrale Banca, Trento, Italy
ettore.battaiola@cassacentrale.it
[2] University of Trento, Trento, Italy
{fabio.massacci,channam.ngo,p.sterlini}@unitn.it

Abstract

Europe has the largest global invoice factoring market for over 1.6BEuro. The purpose of a factoring market is to address delay in payments of commercial invoices by buyers of good and services: sellers bring their still-to-be-paid invoices to financial organizations (factors) which provides an advance payment.

The market further growth is hampered by a number of security issues such as the impossibility of factors to check whether an invoice has been already factored (double pledging). A global organization collecting all factored invoices could be a solution but all stakeholders (banks, factors, etc.) have various reasons to *not* wanting to share such data.

In this scenario a distributed, blockchain-based implementation is the only way forward, We describe the security requirements and all key operations for a secure, fully distributed Invoice Factoring Market, hereafter referred to as simply the 'Market'.

Our distributed, asynchronous protocol simulates the centralized functionality under the assumption of the availability of a distributed ledger. We consider security with abort (in absence of honest majority).

1 Introduction

When a seller issues a commercial invoice to a buyer, the latter may have several months to pay it (from 30days to 120days). In the meanwhile, the seller has to pay its own suppliers, its workers and other operational costs. To cope with such payments delays by buyers, the seller brings the invoice to a factor (which might be a bank or another specialized financial organization) and try to obtain an advance payment by pledging the invoice as a collateral. The factor grants a payment, typically a fraction of the amount on the invoice, subject to a risk assessment involving the credit worthiness of both buyer and seller [26].

Europe has the world largest factoring market and the volume of factored invoices has increased to 1,6BEuro in 2017 from less than a billion in 2010. This increase can partially be explained by several factors: economic growth and therefore a larger number of invoices, increased ability for factors to estimate risk, lower cost of capital and entrance of new actors

*This work has been partly supported by the European Union through the European Institute of Innovation and Technology - EIT Digital under Task 18213 UNBIAS in the 2018 Action Plan (UNITN was involved in: Task 18213-A1802 - Activation of Community/Pilot preparation; Task 18213-A1803 - Design of security and privacy mechanisms; and Task 18213-A1804 - Platform development). We would like to thank R. Hoeksma, G. Kuper, R. Rajani and other members of the UNBIAS team for many useful discussions. Order of authors is alphabetical. EB provided a broad overview of the factoring business in Europe and Italy. FM and PS captured the business requirements for the protocol and its ideal functionality, FM and CNN devised the security protocol. Cassa Centrale Banca and the European Union are not responsible for any claim stated in this paper which remains with the authors.

in the market. Yet, most small companies are poorly served by this financial market which is instead recognized to have the potential to address the liquidity needs of small companies [16].

Two critical features of an invoice market are extremely difficult to evaluate for factors: authenticated identities of parties involved (seller and buyer) and whether or not an invoice has already been factored. As a result, factors can be victim of frauds by sellers that factor non-existent invoices or factor an invoice to multiple factors (*double pledging*). At the same time, buyers might be contacted by fraudulent factors that claims to have factored an invoice of a legitimate seller and demand payments for such invoices. Even in absence of frauds, paying the invoice to wrong organization creates a number of issues.

A quick solution would have been a centralized system where all invoices could be stored and checked by interested parties. Yet, in spite of the existence of the factoring market for centuries, no such entities emerged[1].

The main reason for such absence is the quest for confidentiality. For different reasons, buyers, factors, and sellers have no interest of sharing information on past invoices as it can be used against them. For example, a buyer knowing that a seller routinely factor its invoices may worsen the payment terms "as you will receive the money anyhow, right?" Another example: a factor, knowing that there is an ongoing factoring relationship on the invoices between a seller and a credit worthy buyer, may come forward to the seller offering better factoring terms than its current factor. Such access control relations cannot be managed by traditional role based access control [8] or attribute based access control [29] for single invoice factoring as they would be extremely dynamic and thus costly to maintain and update [13].

To address such challenges, we propose an ideal reactive functionality where all the parties send their private inputs to a trusted third party, which manages the market. Then we show how this information can be captured by using a *distributed ledger with commitments* that protects the integrity of the data and the confidentiality of all interested parties.

Our goal is to mimic as much as possible the actual current process and minimize the role of cryptography and the blockchain ledger to boost the possibility of adoption by existing automated systems for invoice managements at sellers, buyers and factors. Our security protocol is hybrid. We make use of a permissioned, distributed ledger, where authenticated peers maintain the consensus for a distributed ledger which stores all commitments of all secret values while the secret values are only stored locally by the peer who owns the data.

A key feature of our protocol is that *access to the data must go through the data owner*, i.e. a peer who doesn't have the desired data must request the data from the peer who owns it. The data, if accessible by the requesting peer, will be sent through a private secure channel between the two peers (we also called this an out-of-band channel). The commitments in the distributed database are used only to verify the received data (and possibly for solving disputes).

2 Key Security and Business Requirements

Advancing payment on invoices (factoring) has long tradition in the banking and financial sector [26], starting from the advance payments by Venetian Merchants well described in Shackespeare. It is the most common way to provide liquidity to companies. The major reason for the existence of such market is that commercial payments are often late (The Netherlands have an average of 40days while Italy arrives to more than 80days and Spain lags behind). The problem is so pervasive that a European Directive has been issued without much effect.

[1] In contrast centralized entities dealing with transactions such as stock, commodities, and futures exist [12] and even their blockchain based equivalent [19].

Intuitively, the entire relationship is essentially based on a "story" (the invoice) by the Seller who told the Factor that the Buyer owed him something. The invoice reports at least

- who are the Seller and the Buyer (typically identified by the VAT number),
- the Seller's credit (the amount) and the deadline when the payment is due,
- the payment's bank (where the Buyer would send the money).

From the perspective of the Factor, it has document in its hand which, if not false, certifies the Seller's credit. Eventually a bank of the Buyer will transfer this amount into the Invoice's bank account and this will happen because the Buyer said to its bank this story was actually true. However, from the perspective of the relationships we have no warranty whatsoever that the artifact supporting the story is actually true, not that the same story has not been told to several different people in the hope of extracting money from all.

Indeed, a seller can issue invoices as long as it wants. It might even be that the invoice is true in the Seller's head but if disputes arises on the debt by the Buyer, the Factor is anyway at zero. It might also be that the Buyer has paid somewhere else and would therefore refuse to pay. While this should be a bad practice, it happens, often by mistakes. For example if the Seller had previous relationship with the Buyer, the Buyer might have automatically used the previous bank account of the Seller without checking whether it has been changed in the meanwhile.

Factoring typically takes two forms: *recourse factoring* and *factoring without recourse*. The former is the most traditional form of factoring, essentially a loan offered by the banking system in which the invoice is simply a speculative form of collateral. The latter is the most common type in Italy and on most advanced financial markets. *Reverse factoring* is a financial (usually exclusive) agreement mostly used by major industrial groups. Reverse factoring allows suppliers to be paid in a timely manner for a fee that is taken care of by the group ("champion"). In this case is the buyer itself that propose factoring to its small, weak, but strategic suppliers, as an instrument in business negotiations to obtain better terms of special products, better terms of delivery, and lower prices. By entering into the supply chain financing mechanism, once the buyer has approved the invoice sent from the seller, the bank, in agreement with the buyer, will provide the seller with advance financing. The buyer then pays the bank at the agreed time, with low interest rate due to its financial strenght and low risk. In this way, small enterprises with bad credit scoring can receive instant payments because "champions can guarantee the bank that the money will be paid. Table1 summarize the differences.

A common approach to factoring is a permanent relationship between a financier and commercial enterprise which is also called *Portfolio factoring* in which *all invoices* from a seller are managed by the factor. In different markets this might have different embodiments and levels of automation.

For example in the Italian system a common way in which this is implemented is the form of RiBa (Ricevuta Bancaria - bill of exchange) which have a high level of automation. The Ri.Ba. as a credit and transfer instrument, is an order addressed by the Seller's bank to the Buyers' banks, requiring to pay a certain sum of money at a fixed time following an interbank mechanism. The creditor's bank receives a set of orders for charges (typically corresponding to invoices) by the Seller. At due time, if the charge is authorized by each Buyer's Bank the money is transferred to the Seller's Bank. In the meanwhile the Seller's bank has already advanced the money of the orders to the Seller.

Remark 1. *The advance payment of the invoice by the Factor may not be the entire amount mentioned in the invoice. This might happen depending on the level of risks and trust relationship between the Factor and the Seller (or the Buyer for non-recourse factoring). In the Italian system an advanced payment based on the invoices brought for a Ri.Ba. might be around*

Table 1: Types of Factoring Mechanisms

Term	Risk Description
Recourse Factoring	The factor advances the payment to the seller but the risk of the buyer failing to pay is born by the seller, who is accountable to the factor in the event of defaulting buyers. The operation may or may not notified to the buyer. The factor has a primary relationship with the seller.
Non-Recourse Factoring	The seller transfers the ownership of the credit to the factor in change for an advance payment. The factor becomes new creditor of the buyer and bears the risk for a defaulting buyer. The transfer is normally notified to buyers at the time in which their credits become due. The factor has a relationship with both seller and buyer.
Indirect (reverse) Factoring	Based on the the line of credit attributable to each seller, the buyer prepares a payment service proposal for the seller in agreement with the factor. The buyer is directly responsible to pay the factor which has a primary relationship with the buyer.

80-100%.

Portfolio factoring makes it more cost-efficient to perform risk assessment, as it is close to impossible for the Seller to double pledge the invoices as they are all managed by its Factor. Yet, it limits the scope of potential customers to those willing to factor all invoices rather than a small selection. The ability to factor single invoices further increases the potential customer base but presents the concrete risk of double pledging.

A Blockchain system could bring several advantages:

1. the certainty of the uniqueness of the invoice,
2. the (potential) acknowledgement of the buyer,
3. the assurance of the advance of that invoice (i.e. it was anticipated by bank x on that day y).

Yet, there is also an important constraint that stems from the PSD2 European Directive. From the viewpoint of the participating banks and payment service providers the transaction must be open and all banks in the blockchain that have a business with the parties involved in the transaction must be potentially able to read it. In other words, *access to the system must be determined in an automatic way by having a relationship with one of the parties of the invoice.*

This challenge is a severe stumbling block that makes Role-Based Access Control [8] as well Attribute-Based-Access Control [29] unsuitable for the purpose as they are both based on *attributes and roles of the subjects* [13] whereas we need to consider the relationship of the subjects with attributes of the object (namely the invoice).

3 Related Works

Distributed Payment Network The most prominent example of a distributed payment network is Bitcoin [22], whose core components are the Proofs-of-Work and the Blockchain. The current bottleneck of Bitcoin is its low throughput in terms of transactions-per-second (TPS). (Roughly, 10 TPS compared to 2000 TPS achieved, e.g. by Visa.) Several variants/extensions of Bitcoin appeared recently, including ZeroCoin [21], ZeroCash [27], and Ethereum [7].

Blockchain-based FinTech Blockchain is widely adopted to implement distributed financial markets (see the survey [4] for a wide range of distributed financial applications) but most of

them store data in clear and the blockchain only provides integrity and fault-tolerance. This allows the network nodes to read the sensitive data even though they cannot alter or contaminate them. As an example DecReg [23] is a private blockchain-based protocol for preventing double-pledging in Invoice Factoring Market. Even though the protocol relies on a Central Authority to regulate the access control to the sensitive data *from outside*, the peers who maintain the network storage *still have access to the stored data*.

Cryptographic Accumulators Cryptographic-based Invoice Factoring Market was introduced as e-Invoice Factoring Problem which is obtainable through Strong Accumulators from Collision-Resistant Hashing [3]. However cryptographic accumulators [6, 25] only allow to prove membership in a finite set hence it addresses only a small part of the Invoice Factoring Market functionality which is the double-pledging problem.

Secure Multiparty Computation (MPC) Seminal feasibility results in the theory of MPC established that any functionality is securely realizable via a distributed protocol in the computational (resp. information-theoretic) setting, assuming honest minority (resp. majority) [28, 10, 2, 5, 24]. The recent progress on efficient implementations of general-purpose MPC protocols (see, e.g., [1]) opened up the way to advanced applications, e.g. to privacy-preserving data mining [18].

4 Ideal Functionality

A typical evolution of the market includes i) **Initialization** of the market and **Onboarding** the participants; ii) **Invoice Registration and Acknowledgement** where a *Seller* and a *Buyer* agree on an invoice's data; iii) **Invoice Factoring and Proposal Acknowledgement** where a Seller and a *Factor* agree on and assign (when the Factor advances the agreed amount) an invoice factoring proposal, and finally iv) the **Payment Registration and Invoice Settlement** where the Buyer of an invoice pays and the corresponding assignee (either the Seller or a Factor - in the case where the invoice is factored) receives the payment.

For the actual transfers of money, all parties rely on a set of trusted parties called *Payment Service Providers*[2] (PSPs) to route the money transfers. In typical embodiment. Those might be banks or other organisations licensed to undertake payment services. In this respect one can use traditional payment systems or novel cryptographic ones. See Massacci et al. [20] for a comprehensive survey and the corresponding functionalities that must be implemented.

Remark 2. *To keep the protocol simple, we omit the operations where a Buyer disagrees with a Seller on an invoice registration or when a Factor disagrees with a Seller on an invoice proposal (after negotiation). Such steps would obviously be present in a full commercial implementation.*

For expository purposes, both in the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$'s and in the protocol Π_{DIF}'s description we allow an adversary to abort the computation after receiving its own intermediate outputs. This flavor of security is known as security with aborts [14]. The protocol can be extended to avoid scot-free aborts using penalties [17] as done in [19] for the futures market.

Initialization and Onboarding. As shown in Fig. 1, an invoice factoring market comprises of a set of sellers $\mathbb{S}$, buyers $\mathbb{B}$, factors $\mathbb{F}$, and the set of *known and trusted* PSPs $\mathbb{P}$. The market needs also to store a set of invoices $\mathbb{I}$, input by sellers, and a set of money transfers $\mathbb{M}$, input

[2] PSPs are also trusted parties like the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$, we might as well integrate their functionality directly into the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$. However, we detach them here for explicit description of the interactions between the PSPs and other parties. The relationship between PSPs themselves is heavily regulated and therefore trusted. The relationship between the creditor and the debtor isn't assured and is the target of our proposal.

Initialization Initially fix the set of recognizable PSPs $\mathbb{P}$; set $t = 0$ and other stored sets as empty: $\mathbb{B} = \mathbb{S} = \mathbb{F} = \mathbb{I} = \mathbb{M} = \emptyset$;

Onboarding Upon receiving:

- (join, buyer:B), add $\mathbb{B} = \mathbb{B} \cup \{B\}$ if $B \notin \mathbb{B}$;
- (join, seller:S), add $\mathbb{S} = \mathbb{S} \cup \{S\}$ if $S \notin \mathbb{S}$;
- (join, factor:F), add $\mathbb{F} = \mathbb{F} \cup \{F\}$ if $F \notin \mathbb{F}$;

Figure 1: Initialization of the invoice factoring market and onboarding parties

only by the PSPs. The market evolves in rounds, at round $t = 0$, the set of PSPs is known while the sellers, buyers and factors sets are empty. A party can onboard into the market by sending its (join, id) to $\mathcal{F}_{\mathsf{CIF}}$.

Remark 3. *PSPs are responsible to certify the correspondence to the protocol actions to some actions in the real world. This corresponds to value creation into the system (See step 1 in [20]), i.e. a PSP announces into the systems that some money transactions have taken place in the real world. Hence it is critical that they are strongly authenticated with an authentication mechanism that ties their identities to identities in the real world [15].*

Buyers and Sellers could also be authenticated but they could still introduce fraud even if authenticated. For example Buyer and Sellers could collude to create bogus invoices to extract money from Factors and disappear. However, the presence of a strong eIDentity mechanisms [15] would diminish the number of frauds and would be anyhow necessary for dispute resolution in the real world.

Invoice Registration and Acknowledgement. As illustrated in Fig. 2, to register an invoice into the $\mathcal{F}_{\mathsf{CIF}}$:

1. The seller S first composes a standardized invoice document D. The seller S must include some key information such as
   - $H(D)$ a hash (e.g. SHA-1) of the readable invoice document, serves as a unique ID of an invoice;
   - S its own ID; and B the ID of the buyer;
   - v the value of the invoice; and e the expiration date of the invoice.

   We also use the notation $map(H, (s_1, \ldots, s_n))$ to indicate the sequence $(H(s_1), \ldots, H(s_n))$.
2. The seller S then can register these information as an invoice I into $\mathcal{F}_{\mathsf{CIF}}$ as well as send the invoice document to the Buyer B. The buyer status of I is now set as $I.\sigma_B = \mathsf{sent}$.
3. The buyer B can now pickup the invoice I, cross-check it with the (out-of-band) received invoice document. If everything is correct, B will acknowledge the invoice I, i.e. the buyer status of I is now set as $I.\sigma_B = \mathsf{acked}$[3].
4. The PSP P for handling payment related to this invoice is also specified during invoice registration since it has to be trusted by the Seller S.

Invoice Factoring and Proposal Acknowledgment. To factor an invoice I, three parties must participate: the seller S, the factor F and the PSP P (Fig. 3).

[3]The B could also abort the invoice registration by sending (abort). As we said, we omit the operation when Buyer dis-acknowledges an invoice for some reason.

Invoice Registration Seller S composes the invoice document D (which is the readable form of the invoice), sends it out-of-band to the buyer B, and register the important information into the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$.

Upon receiving (send, $H(D)$, S, B, v, e, P) from $S \in \mathbb{S}$, the functionality $\mathcal{F}_{\mathsf{CIF}}$ will

1. create I and add $\mathbb{I} = \mathbb{I} \cup I$;
2. set the status $I.\sigma \equiv (I.hash, I.sell, I.buy, I.val, I.exp, I.pay)$ to $(H(D), S, B, v, e, P)$;
3. append $I.\sigma_B = \mathsf{sent}$ into I;
4. send (send, S, $H(I, map(H, I.\sigma))$) to all parties;
5. send (send, I, $I.\sigma$) to $I.sell$ and $I.buy$;
6. send (send, I, $map(H, I.\sigma)$, $I.sell$, $I.val$) to $I.pay$.

Invoice Acknowledgement B upon receiving (send, I,$I.\sigma$), cross-checks the invoice document D, and I, if they are consistent, sends (ack, $H(I)$) to $\mathcal{F}_{\mathsf{CIF}}$,

Upon receiving ($\mathsf{ack}, H(I)$) from $B \in \mathbb{B}$, the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$ will

1. retrieve I from $\mathbb{I}$;
2. update $I.\sigma_B = \mathsf{acked}$;
3. send (ack, B, $H(I, map(H, I.\sigma))$) to all parties;

Figure 2: Invoice registration and acknowledgment

1. S and F must first negotiate out-of-band the factoring term, i.e. the advanced amount $I.amount$, which requires S to send the invoice document to the factor F.
2. The seller then can make an invoice proposal and register it into the ideal functionality, i.e. the factor is updated as $I.factor = F$, the advanced amount is set as $I.amount = v$, and the factor status is now $I.\sigma_F = \mathsf{factor\_sent}$.
3. The factor will now be able to retrieve I from the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$ and cross-checks the data for consistency, then either acknowledge (by sending $\mathsf{ack\_factor}$, where the factor status will be changed to $I.\sigma_F = \mathsf{factored}$)[4].
4. The PSP P will also be notified by the ideal functionality of the factoring situation for it to handle payment correctly. The PSP P is also responsible for updating the factor status to $I.\sigma_F = \mathsf{fpaid}$ when it receives the correct payment from F.

Payment Registration and Invoice Settlement. The final function of the ideal functionality is payment registration and invoice settlement (Fig. 4). As the round t advances, the invoice expiration notification is sent to all related parties ($I.sell$, $I.buy$, $I.factor$ and $I.pay$).

1. The payment registration starts with a payer (could be the buyer B when s/he pays the invoice or the factor F to advance the payment of the invoice) out-of-band comes to a PSP and make a payment with an $I.hash$ as the reference.
2. The PSP checks if the paying amount is consistent with the payer and the expected amount, then routes the payment to the corresponding payee of the invoice:
   (a) If the payer is the buyer B, the amount must be at least $I.val$; and the payee will be the seller S (invoice is not factored) or factor F (if invoice was successfully factored);

[4]Similar to the invoice acknowledgment, we omit the operation where the Factor or the Seller withdraw the proposal.

Invoice Proposal Registration Upon receiving (factor, F, $I.amount$, $I.hash$) from $S \in \mathbb{S}$ the ideal functionality will

1. retrieve I from $\mathbb{I}$;
2. update $I.factor = F$, $I.amount = v$, $I.\sigma_F =$ factor_sent;
3. send (factor, S, $H(I, map(H, I.\sigma))$) to all parties;
4. send (factor, I, $I.\sigma$) to $I.sell$ and $I.factor$;
5. send (factor, I, $map(H, I.\sigma)$, $I.factor$, $I.amount$) to $I.pay$;

Invoice Proposal Acknowledgement The Factor F upon receiving (factor, I), cross-checks the invoice document and I, if they are consistent, sends (ack_factor, I) to $\mathcal{F}_{\mathsf{CIF}}$;

Upon receiving (ack_factor, I) from $F \in \mathbb{F}$:

1. retrieve I from $\mathbb{I}$;
2. update $I.\sigma_F =$ factored;
3. send (ack_factor, F, $H(I, map(H, I.\sigma))$) to all parties;
4. send (ack_factor, I, $I.\sigma$) to $I.sell$ and $I.factor$;
5. send (ack_factor, I, $map(H, I.\sigma)$,$I.factor$, $I.amount$) to $I.pay$;

Figure 3: Invoice Factoring

Payment Registration Upon receiving (pay, $I.hash$) from $P \in \mathbb{P}$:

1. retrieve I from $\mathbb{I}$;
2. if $I.\sigma_F =$ factored, update $I.\sigma_F =$ fpaid, send (pay, $I.\sigma_F$) to $I.sell$, $I.factor$ and P.
3. if $I.\sigma_B \in \{$acked, expired$\}$, update $= .\sigma_B$bpaid, send (pay, $I.\sigma_B$) to $I.sell$ (or $I.factor$ if $I.\sigma_F =$ fpaid), $I.buy$ and P.
4. send (pay, P) to all parties;

Figure 4: Payment registration and invoice settlement

(b) If the payer is the factor F, the amount must be at least $I.amount$.

3. The PSP then registers the payment into the ideal functionality where the new buyer/ factor status of the invoice will be updated accordingly.
4. The related parties will also be notified of the new status.

Security of $\mathcal{F}_{\mathsf{CIF}}$. The ideal functionality $\mathcal{F}_{\mathsf{CIF}}$ clearly captures the required data access control summarized in Table 2 for an invoice factoring market that integrates the trusted PSPs.

It also captures an important functionality for payment system. Every participant must be able to show that it has performed a due diligence process on the data that it has received from the ideal functionality.

Table 2: Invoice Factoring Market Read/Update Permission

Field/Actor	S	S^*	B	B^*	F	F^*F^2	P	P^*	P_F
Permission	*Read Invoice Data*								
$I.hash$	y		y		y	y	y		y
$I.sell \equiv S$	y		y		y	y	y		y
$I.buy \equiv B$	y		y		y	y	y		
$I.factor \equiv F$	y				y				y
$I.pay \equiv P$	y		y		y	y	y		y
$I.val$	y		y		y	y	y		
$I.exp$	y		y		y	y	y		
$I.amount$	y				y				y
$I.\sigma_B$	y		y		y	y			
$I.\sigma_F$	y				y	y			
Permission	*Update Invoice Data*								
$I.\sigma_B$			y				y		
$I.\sigma_F$					y		y		

On the side we provide the access control matrix for an invoice I and the generic Seller $S^* \neq I.sell$, Buyer $B^* \neq I.buy$, Factor $F^* \neq I.factor$ and PSP $P^* \neq I.pay$ and P_F which is the PSP of the Factor $F = I.factor$. We also have Factor F^2 which is also different from $I.factor$ but has received from the seller the information on the invoice I for a quote. As one can immediately see, only Dynamic ABAC can set appropriately this access control matrix.

5 Cryptographic Building Blocks for the Committment-based Scheme

Distributed Ledgers are ledgers maintained by a network of nodes. The most important property, e.g. for distributed payment networks, is consensus among the nodes, while still being fully decentralized. We use a distributed ledger, e.g. HyperLedger in PBFT mode (https://www.hyperledger.org) as a byzantine fault tolerant storage for each protocol step, i.e. each broadcast is replaced with a write into the distributed ledger. While we mention HyperLedger in our implementation, we can replace any sub-protocol with other protocols for the same task, without affecting security.

Digital Signature Scheme is a tuple of polynomial time algorithms $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ that runs as follows.

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\lambda)$ takes as input the security parameter λ and returns a pair of signing key sk and verifying key vk.

$\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, M)$ takes as input the signing key sk and the message M, outputs a signature σ.

$\{0,1\} = \mathsf{Vf}(\mathsf{vk}, M, \sigma)$ takes as input the verifying key vk, the message M and the signature σ, return 1 upon successful signature verification and 0 otherwise.

For the formal definitions of digital signature we refer readers to [11].

Commitment Schemes. We rely on a non-interactive commitment scheme Com, with domain $\{0,1\}^*$. We typically write $[\![v]\!] := \mathsf{Com}(v; r_v)$ for a commitment to value v using randomness $r_v \in \{0,1\}^*$. To open a given commitment $[\![v]\!]$, it suffices to reveal (v, r_v), so that a verifier can check that $[\![v]\!] = \mathsf{Com}(v; r_v)$. For the proof of security we need that $[\![v]\!]$ statistically hides the committed value v, and after publishing $[\![v]\!]$ it is computationally infeasible to open the commitment in two different ways. We follow [9] for the formal definitions.

Initialization All parties initialize $t = 0$, and all the empty sets:

1. The set of invoices $\mathbb{I} = \emptyset$;
2. The set of payments transfers $\mathbb{M} = \emptyset$;

All parties accept a set of verifying keys for the PSPs: $\mathbb{P} = \{P.\mathsf{vk}\}$

Onboarding Each party i runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\lambda)$ then broadcasts vk to all parties. Other parties stores vk associated with the party ID.

Figure 5: Initialization and onboarding parties into the protocol Π_{DIF}

6 Protocol Construction

In this section we describe the protocol Π_{DIF} that securely realizes the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$. The security of Π_{DIF} will also be sketched accordingly.

Initialization and Onboarding. The protocol initialization is as simple as all parties set the initial round $t = 0$ and the set of invoices $\mathbb{I}$ and payments $\mathbb{M}$ to be initially empty. The set of verifying keys of the PSPs is also fixed at the beginning of the protocol $\mathbb{P} = \{P.\mathsf{vk}\}$[5]. To onboard a party, it requires that party to generate a pair of signing key sk and verifying key vk. The party then simply broadcasts the verifying key vk to be stored by all other parties. The initialization is illustrated in Fig. 5.

Invoice Registration and Acknowledgement. As shown in Fig. 6, to register an invoice into the protocol:

1. The seller S first does the same as in the ideal functionality, i.e. composes a standardized invoice document D, extracts all the important information $I.\sigma = (I.hash, I.sell\ I.buy, I.val, I.exp)$ from the invoice document. We recycle the notation $map(\mathsf{Com}, (s_1, \ldots, s_n))$ to indicate the sequence $([\![s_1]\!], \ldots, [\![s_n]\!])$. The information is then committed into $map(\mathsf{Com}, I.\sigma)$ and $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$.
2. The initial buyer status $I.\sigma_B = \mathsf{sent}$ is also committed into $[\![I.\sigma_B]\!]$.
3. The seller S then broadcasts $[\![I.\sigma_B]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ and only sends the private data to the buyer B.
4. The PSP P also receives $map(\mathsf{Com}, I.\sigma)$, $I.sell$, $I.val$ from the seller S.
5. The buyer B signs the invoice $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ with sk to obtains the signature σI, updates the buyer status to acked (to be committed to $[\![I.\sigma_B = \mathsf{acked}]\!]$).
6. The buyer B sends the updated status $I.\sigma_B = \mathsf{acked}$ to seller S then broadcasts the tuple $(\sigma I, [\![I.\sigma_B = \mathsf{acked}]\!])$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$.
7. The other parties verify the signature before accepting the data.
8. Finally seller S forwards σI and $I.\sigma_B = \mathsf{acked}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to factor F if I is factored to F.

We note that (1) whenever a party sends some private data to another party in a step in the protocol, the corresponding randomness used for the commitment of that private data is also included. Hence we omit the randomness in the protocol description for simplicity; (2) whenever a party receives some private data from another party, a check for data consistency

[5] We assume the PSPs have run key generation and obtain a pair of signing key $P.\mathsf{sk}$ and verifying key $P.\mathsf{vk}$ before the protocol starts.

Invoice Registration Seller S

1. extracts invoice details, i.e. $I.hash$, $I.sell$ $I.buy$, $I.val$, $I.exp$, from the invoice document D;
2. commits $map(\mathsf{Com}, I.\sigma)$;
3. commits and broadcasts $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$;
4. commits and broadcasts $[\![I.\sigma_B = \mathsf{sent}]\!]$;
5. sends $I.\sigma$, $map(\mathsf{Com}, I.\sigma)$, the corresponding invoice document D and $I.\sigma_B = \mathsf{sent}$ to the Buyer B;
6. sends $map(\mathsf{Com}, I.\sigma)$, $I.sell$, $I.val$ to PSP P;

Invoice Acknowledgement Buyer B upon receiving the private data from the seller S;

1. signs and broadcasts $\sigma I = \mathsf{Sig}(\mathsf{sk}, [\![I, map(\mathsf{Com}, I.\sigma)]\!])$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$;
2. update $I.\sigma_B = \mathsf{acked}$;
3. commits and broadcasts $[\![I.\sigma_B = \mathsf{acked}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$;
4. sends $I.\sigma_B = \mathsf{acked}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to Seller S;

Seller S forwards σI and $I.\sigma_B = \mathsf{acked}$ together with $I.hash$ to factor F if I is factored to F.

Figure 6: Invoice Registration and Acknowledgment in the distributed protocol Π_{DIF}

between the private data and the public commitments must be carried out; and (3) whenever a signature verification or a check regarding the data consistency fails, the complaining party open the private data to show the inconsistency and the protocol is aborted[6].

Invoice Factoring and Proposal Acknowledgment. We illustrates the protocol steps to factor an invoice in Fig. 7.

1. As in the ideal functionality, S and F must first negotiate out-of- band the factoring term, i.e. the advanced amount $I.amount$. The out-of-band negotiation requires S to send the invoice document D to the factor F.
2. The seller S then can register an invoice proposal into the protocol by committing and broadcasting $[\![I.factor = F]\!]$, $[\![I.amount = v]\!]$, $[\![I.\sigma_F = \mathsf{factor\_sent}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$.
3. As usual, S forwards the private data $I.\sigma$, $I.\sigma_B$, $I.factor$, $I.amount$ and $I.\sigma_F$ to F.
4. The other parties only accept the new proposal if there is no previous proposal, i.e. there is no tuple $([\![I.factor]\!], [\![I.amount]\!], [\![I.\sigma_F]\!])$ for $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$.
5. The PSP P is also involved in this step:
   (a) It is required to generate a signature $\sigma' I$ on the invoice proposal $[\![I.hash, I.factor, v]\!]$.
   (b) P then sends the signature $\sigma' I$ to S who will forward it to F.
   (c) The factor F then can acknowledge the proposal similarly to the invoice acknowledgment, i.e. commits and broadcasts $[\![I.\sigma_F = \mathsf{factored}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ while sending $I.\sigma_F = \mathsf{factored}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to Seller S and the PSP P.

Payment Registration and Invoice Settlement. The protocol steps for payment handling is described in Fig. 8. The invoice expiration is moderated by the current assignee of the invoice, let it be the seller S or the factor F, by committing and broadcasting $[\![I.\sigma_B = \mathsf{expired}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$. The private data is sent to the buyer B and the PSP P is also notified. The PSP P has an important role in finalizing invoice proposals and invoice settlements.

1. As in the ideal functionality, P receives payment out-of-band from a payer and processes the payment with the stored information.

[6]The private data and the public commitments can be then used for post-protocol dispute resolution.

Invoice Proposal Registration Seller S

1. extracts the factoring term, i.e. v, from the agreement with the factor F;
2. commits and broadcasts $[\![I.factor = F]\!]$, $[\![I.amount = v]\!]$, $[\![I.\sigma_F = \mathsf{factor\_sent}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$;
3. sends $I.factor = F$, $I.amount = v$, $I.\sigma_F = \mathsf{factor\_sent}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to factor F;
4. sends $map(\mathsf{Com}, I.\sigma)$, $I.factor = F$, $I.amount = v$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to PSP P
5. obtains $\sigma' I = \mathsf{Sig}(\mathsf{sk}, [\![[\![I.hash]\!]|I.factor|v]\!])$ from PSP P and forwards it to factor F;

All parties reject the proposal if there is already a tuple $([\![I.factor]\!], [\![I.amount]\!], [\![I.\sigma_F]\!])$ for $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$.
Factor F checks $1 = \mathsf{Vf}(\mathsf{vk}, \sigma' I, [\![[\![I.hash]\!]|I.factor|v]\!])$.

Invoice Proposal Acknowlegdement Factor F upon receiving the private data from Seller S

1. update $I.\sigma_F = \mathsf{factored}$;
2. commits and broadcasts $[\![I.\sigma_F = \mathsf{factored}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$;
3. sends $I.\sigma_F = \mathsf{factored}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to Seller S;
4. sends $map(\mathsf{Com}, I.\sigma)$, $I.factor = F$, $I.amount = v$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to PSP P;

Figure 7: Invoice Factoring in the distributed protocol Π_{DIF}

Invoice Expiration Seller S (or Factor F) can check for one's currently assigned invoice I at every tick t:

1. if $I.exp > t$, update $I.\sigma_B = \mathsf{expired}$;
2. commits and broadcasts $[\![I.\sigma_B = \mathsf{expired}]\!]$ together with $[\![I.hash]\!]$;
3. sends $I.\sigma_B = \mathsf{expired}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to Buyer i;
4. sends $I.hash$, $I.sell$ (or $I.factor$) together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to PSP P.

Payment Registration The payment service provider P will

1. extracts the payment details, i.e. pr, v, $[\![I.hash]\!]$ and find the data related to $[\![I.hash]\!]$;
2. if $I.\sigma_F = \mathsf{factored}$ (the Factor is paying according to the factoring term):
   (a) check $pr = I.factor$, $v \geq I.amount$;
   (b) update $I.\sigma_F = \mathsf{fpaid}$ and route payment to $I.sell$;
   (c) commits and broadcasts $[\![I.\sigma_F = \mathsf{fpaid}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$
   (d) sends $I.\sigma_F = \mathsf{fpaid}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to Seller S and Factor F;
3. else if $v \geq I.val$ (the Buyer is paying the invoice, could be after the expiration date):
   (a) update $= .\sigma_B \mathsf{bpaid}$ and route payment to current assignee of $[\![I.hash]\!]$;
   (b) commits and broadcasts $[\![I.\sigma_B = \mathsf{bpaid}]\!]$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$;
   (c) sends $I.\sigma_B = \mathsf{bpaid}$ together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$ to current assignee of $[\![I.hash]\!]$;
4. otherwise simply reject the payment.

Figure 8: Finalizing Invoice Proposal and Invoice Settlement in Π_{DIF}

2. P then registers the payment into the protocol by committing and broadcasting $[\![I.\sigma_F = \mathsf{fpaid}]\!]$ to finalize an invoice proposal (or $[\![I.\sigma_B = \mathsf{bpaid}]\!]$ to settle an invoice) together with $[\![I, map(\mathsf{Com}, I.\sigma)]\!]$.
3. The private data is forwarded to the involved parties, i.e. the seller S (in case of invoice settlement and the invoice is not factored), the factor F (in case of invoice proposal finalization or the factor has been assigned the invoice), the buyer B (in case of invoice settlement).

6.1 Security Analysis (Sketch)

We sketch here the key step of the security proof. As in standard simulation-based security proofs, we must exhibit an efficient simulator interacting with the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$ that is able to fake the view of any efficient adversaries corrupting a subset I of the untrusted parties in an execution of protocol Π_{DIF}.

On a very high level, our simulator $\mathcal{S}$ works as follows. During the simulation, it commits to zero values for each commitment forwarded by an honest party in the real protocol and relies on the ideal functionality $\mathcal{F}_{\mathsf{CIF}}$ to register, acknowledge invoices, invoice proposals and payments.

The hiding property of the commitment scheme implies that the above simulation is indistinguishable to the view generated in a mental experiment where the simulator $\mathcal{S}$ is given the real inputs corresponding to each honest party. The only difference between this mental experiment and a real protocol execution is that in the former experiment the market evolves using the data held at the beginning by each corrupted party, whereas in the latter experiment the adversary can try to cheat and fake the data of a corrupted party (e.g., by committing or sending wrong value). However, the binding property of the commitment scheme and the security of the digital signature scheme ensure that such cheating attempts only succeed with a negligible probability.

This allows us to conclude that the view simulated in the ideal world (with the functionality $\mathcal{F}_{\mathsf{CIF}}$) is computationally indistinguishable from the view in a real execution of the protocol, thus establishing the security of Π_{DIF}.

7 Concrete Architectural Implementation

We report here on the proposed implementation in the UNBIAS project[7].

The ultimate goal of this network is to digitize and digitalize the invoicing process amongst buyers and sellers. UNBIAS defines following roles: Seller, Buyer, Payment Switch/PSP, Factor.

The first three roles are what we see as the core roles of the network and they form the basis on which the UNBIAS initiative is built. Factor is the role which will help achieve the objective of the UNBIAS is that is to enable easy access to funds for SMEs. There are more roles possible in this network and would be considered to be added in future.

The Distributed ledger is hosted by the Peer Nodes (Dense Compute) and each peer node has copy of ledger locally. Peer nodes are the nodes which do the heavy tasks like doing consensus and verifying commitments. Since these tasks are quite resource intensive, the systems running the node have to be high configuration hardware to support the load. Also only these nodes have the copy of ledger so eventually it would need high amount of storage preferably with high performance.

Apache Kafka is the consensus protocol for Hyperledger fabric. Kafka is primarily a distributed, horizontally-scalable, fault-tolerant, commit log. A commit log is basically a data structure that only appends. No modification or deletion is possible, which leads to no read-/write locks, and the worst case complexity O(1). There can be multiple Kafka nodes in the blockchain network, with their corresponding Zookeeper ensemble. Verification of commitment is necessary to trust the validity of the invoice transactions reported on the ledger. An hash has to be verified against the hashes in the ledger and its chain to the root it is an resource intensive task. Hence it should be run on peer node.

8 Conclusion

The market for invoice factoring can significantly benefit from the deployment of distributed ledger solution. A blockchain would make it possible for factors to check whether an invoice has been already factored (double pledging) without forcing all stakeholders (banks, factors, etc.) to share and disclose their data.

[7] https://unbias.eu/.

In this paper we have described the security requirements and all key operations for a secure, fully distributed Invoice Factoring Market. Our distributed, asynchronous protocol simulates the centralized functionality under the assumption of the availability of a distributed ledger and offers security with abort (in absence of a honest majority).

Looking at the first results from the introduction of the Italian obligation of electronic invoicing[8], there seems to be a reduced propensity of Italian companies to rely on their bank(s) as a provider of services for electronic invoicing. In an economic system characterized by multi-banked companies[9], the tight bond with a single bank is experienced more as a penalty than as a real advantage and companies prefer the use of intermediaries specialized in electronic invoicing. Given this scenario, then the adoption of electronic invoicing could only be an incentive for the development of a blockchain dedicated to the certification of the uniqueness and source of the invoices, without restricting the potential scope of the blockchain itself.

References

[1] David W Archer, Dan Bogdanov, Benny Pinkas, and Pille Pullonen. Maturity and performance of programmable secure computation. *Proc. of IEEE S&P*, (5):48–56, 2016.

[2] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of ACM STOC*, pages 1–10. ACM, 1988.

[3] Philippe Camacho, Alejandro Hevia, Marcos Kiwi, and Roberto Opazo. Strong accumulators from collision-resistant hashing. In *Proc. of ISC*, pages 471–486. Springer, 2008.

[4] Fran Casino, Thomas K Dasaklis, and Constantinos Patsakis. A systematic literature review of blockchain-based applications: current status, classification and open issues. *Telematics and Informatics*, 2018.

[5] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.

[6] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *Cryptographers' Track at the RSA Conference*, pages 127–144. Springer, 2015.

[7] Ethereum. A next-generation smart contract and decentralized application platform, 2015.

[8] David Ferraiolo, Janet Cugini, and D Richard Kuhn. Role-based access control (rbac): Features and motivations. In *Proc. of ACSAC*, pages 241–48, 1995.

[9] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[10] Shafi Goldwasser. How to play any mental game, or a completeness theorem for protocols with an honest majority. *Proc. of ACM STOC*, pages 218–229, 1987.

[11] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[12] Larry Harris. *Trading and exchanges: Market microstructure for practitioners*. Oxford University Press, 2003.

[13] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162), 2013.

[8] http://www.gazzettaufficiale.it/eli/id/2018/10/23/18G00151/sg.

[9] An Italian company may use on average the services of three-four banks and some companies more than a dozen.

[14] Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On achieving the "best of both worlds" in secure multiparty computation. *SIAM journal on computing*, 40(1):122–141, 2011.

[15] Gerhard Jakisch. E-signature versus e-identity: the creation of the digital citizen. In *Proc. of DEXA*, pages 312–316. IEEE, 2000.

[16] Leora Klapper. *The role of factoring for financing small and medium enterprises*. The World Bank, 2005.

[17] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Proc. of IEEE S&P*, pages 839–858, 2016.

[18] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI Global, 2005.

[19] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams. Futuresmex: Secure, distributed futures market exchange. In *Proc. of IEEE S&P*, pages 453–471, 2018.

[20] Fabio Massacci, Chan-Nam Ngo, and Julian M Williams. Decentralized transaction clearing beyond blockchains. 2016. Available at SSRN: https://ssrn.com/abstract=2794913.

[21] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proc. of IEEE S&P*, pages 397–411. IEEE, 2013.

[22] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[23] Joris Oudejans, Zekeriya Erkin, et al. Decreg: A framework for preventing double-financing using blockchain technology. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 29–34. ACM, 2017.

[24] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of ACM STOC*, pages 73–85. ACM, 1989.

[25] Leonid Reyzin and Sophia Yakoubov. Efficient asynchronous accumulators for distributed pki. In *International Conference on Security and Cryptography for Networks*, pages 292–309. Springer, 2016.

[26] Freddy R Salinger. *Factoring: the law and practice of invoice finance*. Sweet & Maxwell, 2006.

[27] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Proc. of IEEE S&P*, pages 459–474. IEEE, 2014.

[28] AC Yao. Protocols for secure computations (extended abstract). in proceedings. In *Proc. of IEEE FOCS*, pages 160–164, 1982.

[29] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *Proc. of IEEE ICWS*. IEEE, 2005.