Fair and Decentralized Exchange of Digital Goods

Ariel Futoransky
Disarmista
futo@disarmista.com

Carlos Sarraute
Wibson & Grandata
charles@grandata.com

Daniel Fernandez
Wibson
daniel@wibson.org

Matias Travizano
Wibson
mat@wibson.org

Ariel Waissbein
Disarmista
wata@disarmista.com

Abstract

We construct a privacy-preserving, distributed and decentralized marketplace where parties can exchange data for tokens. In this market, buyers and sellers make transactions in a blockchain and interact with a third party, called notary, who has the ability to vouch for the authenticity and integrity of the data. We introduce a protocol for the data-token exchange where neither party gains more information than what it is paying for, and the exchange is fair: either both parties gets the other's item or neither does. No third party involvement is required after setup, and no dispute resolution is needed.

1 Introduction

Fair exchange, smart contracts and contingent payments have occupied the cryptographic community for decades. They relate to problems where parties exchange a piece of data or a service, for data or payment. We look here upon the problem of selling personal data.

The importance of privacy in digital transactions continues to grow in these times, when marketing is ubiquitous and based on the footprint of digitalized lives. Individuals are willing to loosen their privacy expectations in exchange for a service (e.g., social networks or webmail services) or even money.

Consider a a distributed and decentralized marketplace where some parties can buy personal data. For example, an apparel company is willing to pay for the 2-weeks browsing history of people who went to Coachella Festival, or a financial institution willing to pay for the last places visited by people between the ages of 20 and 35, earning over $150K, in New York City. We are interested in (i) a question that can be encoded in a predicate that we call audience matching criteria, and (ii) a payload, or piece of information, tied to this predicate.

In this market, a *buyer* publishes an offer that includes an audience match criterion, a payload description, and an amount he is willing to pay for an answer. A trusted third party

creates certificates for the sellers, wherein a certificate contains an authenticated answer for an offer (e.g., authenticated audience match value and payload). Each seller receives certificates, and when he receives an offer, he interacts with the buyer to evaluate the audience criterion. When there is a match, the buyer starts a contract in a blockchain and the seller closes the contract. With an atomic swap, tokens are exchanged for the data. This trusted third party can be materialized as a bank, who can certify questions about spending and financial status; or a telecommunications company, which can certify the geographical location of its subscribers (via triangulation with the antennas their cell phones connect to).

Contributions and Related Work

We introduce a decentralized and distributed marketplace for selling personal digital data. The market design we present was used as a blueprint for a real construct, and the abstractions adhere to realistic security problems.

We design an UC-ideal functionality which implements secure exchanges, and a real life protocol that securely realizes this. This protocol is privacy friendly: it hides data and transactions from the public. If parties are honest, then the buyer, and only the buyer, can access this data and the seller gets paid.

The protocol and marketplace formalize an existing data marketplace named Wibson (Travizano et al., 2018; Fernandez et al., 2020). The secure exchange mechanism was presented in Futoransky et al. (2019). This marketplace uses an Ethereum side-chain, and a gas efficient protocol named BatPay for the recurrent micropayment of tokens (Mayer et al., 2020). Users connect through a mobile app in their phones, and check a message board where buyers post their offers. The solution herein needs to conforms with a high standard both in terms of the security and privacy, and in terms of the computational and communication costs. Additionally, this marketplace features a cryptographic primitive called WibsonTree, designed to preserve users' privacy by allowing them to demonstrate predicates on their personal attributes, without revealing the values of those attributes (Futoransky et al., 2020).

In a fair exchange, two parties wish to exchange a piece of data each holds and is secret to the other. While a result by Cleve (1986) shows that basic fair exchange is impossible in a 2-party setting, modifications and restrictions demonstrate that it is achievable (Asokan et al., 2000; Micali, 2003; Okada et al., 2008; Campanelli et al., 2017).

The setting of the paper differs from the models underlying these articles. Campanelli et al. (2017) uses a blockchain to perform atomic swaps, but moves to use this for providing services, such as Sudoku-puzzle solving through ZK-proofs.

2 A Model for Decentralized Exchange of Digital Goods

We use the Universal Composable Security framework (Canetti, 2001) to formalize the notion of a marketplace. We use the $\mathcal{F}_{\mathtt{ca}}$–hybrid model, where parties are allowed to register and retrieve public keys from a certification authority (Canetti, 2003). We use the plain communication model with unauthenticated asynchronous communication, without guaranteed delivery and with possible replay (see for example Canetti (2000)). Furthermore, we restrict attacks to static Byzantine corruptions.

The protocol is played by two main parties, a seller $\mathcal{S}$ and a buyer $\mathcal{B}$. A third party, the notary $\mathcal{N}$, vouches for the authenticity of data. That is, the notary may provide a seller with a 'certificate' which includes data and an audience match value *binded* to the seller's id. A fourth party, $\mathcal{W}$, plays the role of a blockchain. It is known (Badertscher et al., 2017; Garay et al., 2015; Kiayias et al., 2015) that not all the properties of a blockchain are captured by an Interactive Turing Machine (ITM). Nonetheless, we model the blockchain as an ITM for the sake of simplicity, capturing some of its properties.

Parties have an account with the blockchain that is binded to their id, i.e., they registered a signature verification public key v with $\mathcal{F}_{\mathsf{ca}}$ and $\mathcal{W}$ has retrieved them all. Furthermore, we assume that the notary $\mathcal{N}$ is known by all parties, i.e., they know that the id $\mathcal{N}$ is associated with a notary that can sign data and this data ought to be trusted. $\mathcal{W}$ stores a table (ledger) where each party id is associated with an amount of tokens the party owns, and publishes updates on each output. Assume that on initialization, $\mathcal{W}$ receives the initial state of the ledger. At any point $\mathcal{W}$ is allowed to make token transfers from one account to the other, or to momentarily immobilize a token. We assume no tokens are created or removed. In order to model the blockchain's ability to multicast messages, we assume that $\mathcal{W}$ writes messages to a 'public' tape, and the adversary may decide deliver it to any party, like it does with any message. Similarly, we assume that buyers multicast messages and the adversary may decide what to do with them.

The blockchain, $\mathcal{W}$, executes a contract: it receives messages of the form

$$(\mathsf{Contract\ Open}, id, \text{'Pay N if } x : \mathsf{H}(x) = X\text{'})$$

and

$$(\mathsf{Contract\ Close}, sid, K).$$

When $\mathcal{W}$ receives a Contract Open–message, if this is the first contact with this id and the sender has N tokens for payment, then $\mathcal{W}$ immobilizes N tokens, and puts the message in its public tape. When receiving a Contract Close–message, $\mathcal{W}$ checks for a stored Contract Open–message sid which has not been closed, he checks if $\mathsf{H}(K) = X$. If equality holds, $\mathcal{W}$ transfers N tokens from sender of the Open message to sender of the Close message, and considers the contract closed. Else, $\mathcal{W}$ ignores the Contract Close-message and wait for a new one.

We assume that we are given a function f defining all audience criteria. For a seller attribute $\mathbf{s}$ and a buyer attribute $\mathbf{b}$, f can be evaluated in polynomial time and $f(\mathbf{s}, \mathbf{b}) = 1$ if, and only if, $\mathbf{s}$ matches the criterion defined by $\mathbf{b}$. We assume that all the negotiation details are encoded in $\mathbf{s}$ and $\mathbf{b}$, including the price offered by the buyer, which for the sake of simplicity, we have fixed at 1 token.

Preliminaries

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and H a collision-resistant hash function. We require that Π offers semantic security and remains safe even if an adversary is given $\mathsf{Hash}(K)$ (cf. Futoransky et al. (2006)). This standard requirement in practice does not follow from the definitions. Nonetheless, this can be attained for example starting from a secure block cipher (E, D) (Boneh and Shoup, 2020), a mode of operation that allows transmitting messages of polinomially-bounded length (with a resulting semantically-

secure symmetric cipher), and with $\mathsf{H} : \{0,1\}^n \to \{0,1\}^k$ the (preimage-resistant) hash function H constructed from Enc (see Black et al. (2002)).

Let $\mathcal{F}_{\mathtt{smt}}$ be the ideal-process secure message transmission functionality of Canetti (2001) and let π_E be a real-life protocol securely realizing $\mathcal{F}_{\mathtt{smt}}$, e.g., the protocol of *op. cit.* constructed out of a public-key encryption scheme that is semantically secure against chosen plaintext attack. We assume the leakage function for $\mathcal{F}_{\mathtt{smt}}$ to provide the adversary with the message header and the size of the secret (non-header) portion of the message. Let $\mathcal{F}_{\mathtt{sig}}$ be the signature ideal functionality of Canetti (2003) and let π_S be a signature scheme which is eu-cma; hence this protocol securely realizes $\mathcal{F}_{\mathtt{sig}}$ (Goldwasser et al., 1988; Canetti, 2003)).

3 The Secure Exchange Functionality

We introduce the (abstract) ideal functionality that enables secure exchanges. The protocol has three steps, during **setup** the notary receives an input $(\mathcal{S}, M, \mathbf{s})$ consisting in the id of a seller $\mathcal{S}$, a piece of data (or message) M, and the seller's attribute $\mathbf{s}$; next comes the **offer** where the buyer advertises an audience criterion through an attribute $\mathbf{b}$; and finally, the seller responds to the offer and the **exchange** takes place.

Functionality $\mathcal{F}_{\mathtt{se}}$

1. Upon receiving $(\mathsf{Certify}, sid, \mathcal{S}, M, \mathbf{s})$ from $\mathcal{N}$ check that $sid = (\mathcal{N}, sid')$ and that no other Certify message was received with the same sid.
   - If $\mathcal{S}$ is uncorrupted, send $(\mathsf{Cert}, sid, \mathcal{S}, |M|, |\mathbf{s}|)$ to the adversary. Upon receiving $(\mathsf{Cert\ Received}, sid)$ from the adversary, send $(\mathsf{Cert\ Received}, sid)$ to $\mathcal{S}$ and store $(sid, \mathcal{S}, M, \mathbf{s})$.
   - If the seller $\mathcal{S}$ has been corrupted, send $(\mathsf{Cert}, sid, \mathcal{S}, M, \mathbf{s})$ to the adversary and store $(sid, \mathcal{S}, M, \mathbf{s})$.
2. Upon receiving an offer $(\mathsf{Buy}, bid, \mathbf{b})$ from a buyer, forward to the adversary. Each time the adversary sends a message $(\mathsf{Buy}, bid, \mathbf{b}, id)$ send $(\mathsf{Buying}, bid, \mathbf{b})$ to party with id id and store the message.
3. Upon receiving $(\mathsf{Seller}, bid, sid)$, retrieve (if any) a stored certificate with id sid, and an offer with id bid. If $f_{\mathbf{s}}(\mathbf{b}) = 1$, send the message $(\mathsf{Seller}, bid, sid)$ to the adversary.
   - Upon receiving $(\mathsf{Contract\ Close}, bid)$ from the adversary, transfer 1 token from $\mathcal{B}$ to $\mathcal{S}$.
   - Upon receiving $(\mathsf{Finished}, bid)$ from the adversary when the $\mathcal{N}$ is uncorrupted, send $(\mathsf{Message}, bid, M)$ to $\mathcal{B}$, or upon receiving $(\mathsf{Finished}, bid, M')$ from the adversary when $\mathcal{N}$ is corrupted, send $(\mathsf{Message}, bid, M')$ to $\mathcal{B}$; then abort.

Figure 1: The ideal functionality $\mathcal{F}_{\mathtt{se}}$.

The ideal functionality shown in Fig. 1 gives the adversary the power to decide who gets the offers, as the adversary controls the communication network, and hence also if the seller's message, which closes the transaction, reaches the blockchain, and if the blockchain's message can reach $\mathcal{B}$ or $\mathcal{S}$.

4 The Secure Exchange Protocol

Let $k \in \mathbb{Z}$. The protocol $\rho_{\mathtt{se}}$ in the $(\mathcal{F}_{\mathtt{smt}}, \mathcal{F}_{\mathtt{sig}}, \mathcal{F}_{\mathtt{ca}})$-hybrid model goes as follows.

1. When the notary $\mathcal{N}$ receives an input $(\mathsf{Certify}, sid, \mathcal{S}, M, \mathbf{s})$, he computes $K \leftarrow \mathsf{Gen}(1^k)$, the ciphertext $C \leftarrow \mathsf{Enc}(K, M)$, the hashes $Y := \mathsf{H}(C), X := \mathsf{H}(K)$, and invokes $\mathcal{F}_{\mathtt{sig}}$ with $(\mathsf{Sign}, sid, (\mathbf{s}, Y, X))$ to receive a signature σ. When done, $\mathcal{N}$ sends $(\mathsf{Cert}, sid, K, M, C, \mathbf{s}, Y, X, \sigma)$ to $\mathcal{S}$ via $\mathcal{F}_{\mathtt{smt}}$.

2. Upon receiving $(\mathsf{Cert}, sid, K, M, C, \mathbf{s}, Y, X, \sigma)$, the seller invokes $\mathcal{F}_{\mathtt{ca}}$ with $(\mathsf{Retrieve}, \mathcal{N})$ and waits for $(\mathsf{Retrieve}, \mathcal{N}, v)$, next he invokes $\mathcal{F}_{\mathtt{sig}}$ with $(\mathsf{Verify}, \mathcal{N}, (\mathbf{s}, Y, X), \sigma, v)$ and waits for an answer. If the signature is valid, $Y = \mathsf{H}(C), X = \mathsf{H}(K)$ and $M = \mathsf{Dec}(K, C)$, he outputs $(\mathsf{Cert\ received}, sid)$.

3. When $\mathcal{B}$ receives $(\mathsf{Buy}, bid, \mathbf{b})$, he multicasts $(\mathsf{Buying}, bid, \mathbf{b})$.

4. Upon receiving $(\mathsf{Buying}, bid, \mathbf{b})$, and if it is the first offer with this bid, a seller stores the message and outputs $(\mathsf{Offer\ received}, bid)$.

5. When the seller receives $(\mathsf{Sell}, sid, bid)$,he looks in his storage for a certificate and an offer with these ids. If he finds them and $f(\mathbf{b}, \mathbf{s}) = 1$, $\mathcal{S}$ sends the message $(\mathsf{Selling}, bid, \mathcal{N}, C, \mathbf{s}, Y, X, \sigma)$ to $\mathcal{B}$ via $\mathcal{F}_{\mathtt{smt}}$ and stores (sid, bid).

6. When the buyer receives a Selling–message, he invokes $\mathcal{F}_{\mathtt{ca}}$ with $(\mathsf{Retrieve}, \mathcal{N})$ waits to receive a tag v, he invokes $\mathcal{F}_{\mathtt{sig}}$ with $(\mathsf{Verify}, \mathcal{N}, (\mathbf{s}, Y, X), \sigma, v)$ waits to receive an affirmative verification, and if this happens checks that $f(\mathbf{b}, \mathbf{s}) = 1$. If anything fails, he ignores. Else, he sends the message $(\mathsf{Contract\ Open}, bid,$ 'Pay if $x : \mathsf{H}(x) = X$') to $\mathcal{W}$.

7. Upon receiving a Contract Open–message, $\mathcal{W}$ checks if the sender has a balance of at least one token and ignores of he does not. Else, $\mathcal{W}$ immobilizes a token from the sender and multicasts the Contract Open–message.

8. Upon reading a Contract Open–message for a bid $\mathcal{S}$ for a stored pair (bid, sid), retreives the key associated to sid and sends $(\mathsf{Contract\ Close}, bid, K)$ to $\mathcal{W}$.

9. Upon receiving $(\mathsf{Contract\ Close}, bid, K)$ from $\mathcal{S}$, the blockchain $\mathcal{W}$ looks for an associated Contract Open–message and ignores if not found or if it is already closed. Else, he computes $\mathsf{H}(K)$. If it agrees with X, then he multicasts $(\mathsf{Contract\ Close}, bid, K)$ in his public tape and transfers the immobilized token to $\mathcal{S}$ and modifies the ledger to reflect this change.

10. Upon reading $(\mathsf{Contract\ Close}, bid, K)$, $\mathcal{B}$ outputs $(\mathsf{Message}, bid, \mathsf{Dec}(K, C))$.

11. Upon reading in the new state of the ledger that he was payed, the seller outputs $(\mathsf{Payment\ received}, bid)$.

5 The Main Result

We now prove that the real-life protocol securely realizes the ideal functionality.

Theorem 1. *Protocol $\pi_{\mathtt{se}}$ securely realizes $\mathcal{F}_{\mathtt{se}}$ in the $\mathcal{F}_{\mathtt{ca}}$–hybrid model.*

Proof. By the UC composition theorem it suffices to show that $\rho_{\mathtt{se}}$ securely realizes functionality $\mathcal{F}_{\mathtt{se}}$ in the $(\mathcal{F}_{\mathtt{smt}}, \mathcal{F}_{\mathtt{sig}}, \mathcal{F}_{\mathtt{ca}})$–hybrid model. Let $\mathcal{A}$ be a $(\mathcal{F}_{\mathtt{sig}}, \mathcal{F}_{\mathtt{smt}}, \mathcal{F}_{\mathtt{ca}})$–hybrid model adversary. We define an ideal-process adversary (e.g., a simulator) $\widetilde{\mathcal{A}}$ such that the execution in the ideal-process model and in the $(\mathcal{F}_{\mathtt{smt}}, \mathcal{F}_{\mathtt{sig}}, \mathcal{F}_{\mathtt{ca}})$-hybrid models are indistinguishable by any environment $\mathcal{E}$. The simulator runs an execution with the environment $\mathcal{E}$ and, in parallel, simulates a virtual copy of the hybrid adversary $\mathcal{A}$. That is, $\widetilde{\mathcal{A}}$ acts as an interface between $\mathcal{A}$ and $\mathcal{E}$ by imitating a copy of a real execution of $\rho_{\mathtt{se}}$ for $\mathcal{A}$, incorporating $\mathcal{E}$'s ideal-model interactions and forwarding $\mathcal{A}$'s messages to $\mathcal{E}$.

Assume no corruptions happens.

When $\widetilde{\mathcal{A}}$ receives $(\mathsf{Cert}, sid, \mathcal{S}, |M|, |\mathbf{s}|)$ from the ideal functionality $\mathcal{F}_{\mathtt{se}}$, $\widetilde{\mathcal{A}}$ computes:

- $K \leftarrow \mathsf{Gen}(1^k)$,
- a random string $\tilde{M} \in \{0,1\}^{|M|}$ of size $|M|$,
- $\tilde{C} \leftarrow \mathsf{Enc}(K, \tilde{M})$,
- the hashes $Y := \mathsf{H}(C), X := \mathsf{H}(K)$,

and simulates $\mathcal{F}_{\mathsf{sig}}$ sending $(\mathsf{Sign}, \mathcal{N}, (\mathsf{s}, Y, X))$ to $\mathcal{A}$.

When $\mathcal{A}$ answers with $(\mathsf{Signature}, \mathcal{N}, (\mathsf{s}, Y, X), \sigma)$, $\widetilde{\mathcal{A}}$ simulates $\mathcal{N}$ sending

$$(\mathsf{Cert}, sid, |(K, M, C, \mathbf{s}, Y, X, \sigma)|)$$

to $\mathcal{S}$ through $\mathcal{F}_{\mathtt{smt}}$.[1] If $\mathcal{A}$ delivers a Cert–message, then $\widetilde{\mathcal{A}}$ simulates $\mathcal{F}_{\mathtt{ca}}$ sending $(\mathsf{Retrieve}, \mathcal{N}, \mathcal{S})$ to $\mathcal{A}$. If the hybrid adversary answers ok, then $\widetilde{\mathcal{A}}$ simulates $\mathcal{F}_{\mathtt{sig}}$ sending $(\mathsf{Verify}, \mathcal{N}, (\mathbf{s}, Y, X), \sigma, v)$ to $\mathcal{A}$ and mimics $\mathcal{F}_{\mathtt{sig}}$ verification algorithm. If the verification checks out, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Cert\ Received}, sid)$ to $\mathcal{F}_{\mathtt{se}}$.

When $\widetilde{\mathcal{A}}$ receives $(\mathsf{Buy}, sid, \mathbf{b})$ from functionality $\mathcal{F}_{\mathtt{se}}$, he simulates $\mathcal{B}$ multicasting $(\mathsf{Buying}, sid, \mathbf{b})$. Next, each time $\mathcal{A}$ delivers the message to a party with id id, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Buying}, sid, \mathbf{b}, id)$ to $\mathcal{F}_{\mathtt{se}}$.

Upon $\widetilde{\mathcal{A}}$ receiving $(\mathsf{Sell}, sid, bid)$ from the ideal functionality, $\widetilde{\mathcal{A}}$ simulates $\mathcal{S}$ sending $(\mathsf{Selling}, bid, C, (\mathbf{s}, Y, X), \sigma)$ to $\mathcal{B}$ through $\mathcal{F}_{\mathtt{smt}}$. Upon $\mathcal{A}$ delivering a Selling–message, the ideal-process adversary simulates $\mathcal{F}_{\mathtt{ca}}$ sending $(\mathsf{Retrieve}, \mathcal{N}, \mathcal{B})$ to the hybrid adversary. Party $\widetilde{\mathcal{A}}$ waits for $\mathcal{A}$ to answer with ok, then $\widetilde{\mathcal{A}}$ simulates $\mathcal{B}$ by sending to $\mathcal{W}$

$$(\mathsf{Contract\ Open}, bid, \text{'Pay if } x : \mathsf{H}(x) = X\text{'})$$

[1] Note that $\widetilde{\mathcal{A}}$ only needs the size of the message and not its contents, to simulate this for $\mathcal{A}$, e.g., he can send a string of 1s of the correct size and $\mathcal{A}$ cannot distinguish one from the other.

and waits for $\mathcal{A}$ to deliver this to $\mathcal{W}$. When this happens, the ideal-process adversary checks if $\mathcal{B}$ has at least one token in its account, and ignores if there is less. Else $\widetilde{\mathcal{A}}$ 'immobilizes' one simulated token, and simulates $\mathcal{W}$ multicasting the Contract Open–message. When $\mathcal{A}$ delivers this message to $\mathcal{S}$, the ideal-process adversary simulates $\mathcal{S}$ sending $(\mathsf{Contract\ Close}, bid, K)$ to $\mathcal{W}$. Upon $\mathcal{A}$ delivering a Contract Close–message with id bid, $\widetilde{\mathcal{A}}$ verifies that $H(K) = X$ and if this happens, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Contract\ Close}, bid)$ to $\mathcal{F}_{\mathtt{se}}$ and simulates $\mathcal{W}$ putting this message in its public tape. Upon the adversary $\mathcal{A}$ delivers this message, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Finished}, bid)$ to the ideal functionality $\mathcal{F}_{\mathtt{se}}$.

In order to conclude that the result follows in the uncorrupted case, first notice that the delays in message delivery are incorporated by simulator so that all outputs are synchronized. Second, note that the hybrid adversary learns the headers:

- (Cert, sid),
- $(\mathsf{Selling}, sid)$,
- the size $|(K, M, \mathbf{s}, C, X, \sigma)|$,
- the size $|(\mathbf{s}, C, X, \sigma)|$.

He also learns the full contents of

- $(\mathsf{Buying}, sid, \mathbf{b})$,
- $(\mathsf{Contract\ Open}, sid,$ 'Pay if $x : \mathsf{H}(x) = X$'$)$,
- $(\mathsf{Contract\ Close}, sid, K)$,
- and who receives the first of these messages.

These are computationally indistinguishable and thus the result follows.

Assume $\mathcal{A}$ corrupts $\mathcal{N}$.

In that case $\widetilde{\mathcal{A}}$ corrupts $\widetilde{\mathcal{N}}$ and has dummy $\widetilde{\mathcal{N}}$ output what $\mathcal{N}$ outputs. When $\widetilde{\mathcal{A}}$ receives $(\mathsf{Cert}, sid, \mathcal{S}, M, \mathbf{s})$ from the ideal functionality $\mathcal{F}_{\mathtt{se}}$, he simulates the environment sending this to the simulated $\mathcal{N}$ as input. Moreover, $\widetilde{\mathcal{A}}$ simulates the other parties for $\mathcal{N}/\mathcal{A}$. In particular, if the simulated $\mathcal{N}/\mathcal{A}$ sends a message

$$(\mathsf{Cert}, sid, K', M', C', \mathbf{s}', Y', X', \sigma')$$

through $\mathcal{F}_{\mathtt{smt}}$ to $\mathcal{S}$, then the ideal-process adversary continues the simulation as in the uncorrupted case.

The corruption of the notary can only influence the certificate sent to the seller, which is the only message sent by the notary. If $\mathcal{N}/\mathcal{A}$ sends a certificate which does not have the correct format, or the signature validation is not passed:

$$Y' \neq \mathsf{H}(C') \ \text{ or } \ X' \neq \mathsf{H}(K') \ \text{ or } \ C' \neq \mathsf{Dec}(K', C'),$$

then the seller ignores the message. Assume then that all these controls are passed. Then the simulator continues as in the uncorrupted case: $\widetilde{\mathcal{A}}$ simulates $\mathcal{W}$ sending the Contract Close–message to $\mathcal{B}$. If the adversary $\mathcal{A}$ delivers this message, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Finished}, bid, M')$ to the ideal functionality $\mathcal{F}_{\mathtt{se}}$. Hence, both the $\mathcal{B}$ and $\widetilde{\mathcal{B}}$ pay for M'.

Assume the hybrid adversary corrupts $\mathcal{S}$.

In this case the simulator corrupts $\widetilde{\mathcal{S}}$. Additionally he has $\widetilde{\mathcal{S}}$ output what $\mathcal{S}$ outputs. When $\widetilde{\mathcal{A}}$ receives $(\mathsf{Cert}, sid, \mathcal{S}, \mathbf{s}, M)$ from the ideal functionality $\mathcal{F}_{\mathtt{se}}$, he proceeds analogously

to the uncorrupted case, changing the fake values for real ones. More precisely, $\widetilde{\mathcal{A}}$ simulates key generation, encryption and hashing with the real values, and then invoking $\mathcal{F}_{\texttt{sig}}$ with $(\mathsf{Sign}, \mathcal{N}, (\mathbf{s}, Y, X))$. Upon obtaining $(\mathsf{Signature}, \mathcal{N}, (\mathbf{s}, Y, X), \sigma)$ from the hybrid adversary, $\widetilde{\mathcal{A}}$ simulates sending

$$(\mathsf{Cert}, sid, K, M, C, \mathbf{s}, Y, X, \sigma)$$

from $\mathcal{N}$ to $\mathcal{S}$ through $\mathcal{F}_{\texttt{smt}}$. (Note that here $\widetilde{\mathcal{A}}$ holds M and s, generates K, computes X, C and σ –same as $\rho_{\texttt{se}}$.) When the hybrid adversary delivers the message, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Cert\ Received}, sid)$ to the ideal functionality $\mathcal{F}_{\texttt{se}}$.

Upon receiving $(\mathsf{Buy}, bid, \mathbf{b})$, $\widetilde{\mathcal{A}}$ continues like in the uncorrupted case.

Upon receiving $(\mathsf{Sell}, sid, bid)$ from $\mathcal{F}_{\texttt{se}}$, $\widetilde{\mathcal{A}}$ simulates $\mathcal{S}$ receiving the same message from the environment. If the corrupted seller, $\mathcal{S}$, sends a message

$$(\mathsf{Selling}, sid, C', \mathbf{s}', Y', X', \sigma')$$

to $\mathcal{B}$ through $\mathcal{F}_{\texttt{smt}}$, the ideal-process adversary simulates $\mathcal{F}_{\texttt{ca}}$ sending

$$(\mathsf{Retrieve}, \mathcal{N}, \mathcal{B})$$

to the hybrid adversary and if $\mathcal{A}$ answers with ok, then $\widetilde{\mathcal{A}}$ simulates $\mathcal{F}_{\texttt{sig}}$ sending $(\mathsf{Verify}, \mathcal{N}, (\mathbf{s}', Y', X'), \sigma', v)$ and ignores if the signature validation is not passed. Else, $\widetilde{\mathcal{A}}$ simulates $\mathcal{B}$ by sending to $\mathcal{W}$

$$(\mathsf{Contract\ Open}, sid, \text{`Pay if } x : \mathsf{H}(x) = X\text{'}).$$

If the hybrid adversary delivers the message, $\widetilde{\mathcal{A}}$ simulates $\mathcal{W}$ writing this message to its public tape. When the hybrid adversary delivers the message to the $\mathcal{S}$, $\widetilde{\mathcal{A}}$ waits for $\mathcal{S}$ to send $(\mathsf{Contract\ Close}, bid, K')$ to $\mathcal{W}$. $\widetilde{\mathcal{A}}$ checks if $K' = K$ and ignores if it does not hold. Else, the ideal-process adversary sends $(\mathsf{Contract\ Close}, sid)$ to $\mathcal{F}_{\texttt{se}}$ and simulates $\mathcal{W}$ writing $(\mathsf{Contract\ Close}, sid, K)$ to its public tape. If $\mathcal{A}$ delivers, $\widetilde{\mathcal{A}}$ sends $(\mathsf{Finished}, sid)$ to the ideal functionality $\mathcal{F}_{\texttt{se}}$. Again, $\mathcal{E}$ can't distinguish the hybrid from ideal-process protocol executions.

The cases where the adversary corrupts $\mathcal{W}$ or multiple parties is left out for the sake of brevity, thus concluding the proof.

□

6 Conclusion

Here we proposed a solution to the problem of trading real-world private information using only cryptographic protocols and a public blockchain to guarantee the fairness of transactions. We described a protocol that we call "Secure Exchange of Digital Goods" (Futoransky et al., 2019) between a data buyer $\mathcal{B}$ and a data seller $\mathcal{S}$. The protocol relies on a trusted third party $\mathcal{N}$, which also plays the role of notary in the context of a decentralized Privacy-Preserving Data Marketplace (dPDM) such as the Wibson Marketplace (Travizano et al., 2018; Fernandez et al., 2020).

This protocol converts the exchange of data into an atomic transaction where two things happen simultaneously:

- The buyer $\mathcal{B}$ gets access to the data, by learning the key that enables him to decrypt C (previously received encrypted data).
- The seller $\mathcal{S}$ gets paid for his data by revealing the key.

There exists some open questions that cannot be addressed completely in this paper, but are worth posing to understand the value of the model. For example, can we modify the protocol so that `s` or `b` are not shared? How can we allow a seller to close a contract if he does not have currency (e.g., Ethereum gas) to pay for the transaction? We have actually developed a solution for this problem in (Mayer et al., 2020).

The implementation of the above protocol implies costs to all intervening parties. Each transaction costs at least the computation of a hash, and if this turns to be expensive, the protocol needs to be optimized. This can be done with the BatPay smart contract (Mayer et al., 2020).

References

Nadarajah Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in communications*, 18(4):593–610, 2000.

Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. Cryptology ePrint Archive, Report 2017/149, 2017. `https://eprint.iacr.org/2017/149`.

John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Annual International Cryptology Conference*, pages 320–335. Springer, 2002.

Dan Boneh and Viktor Shoup. *A Graduate Course in Applied Cryptography*. 2020. Published online, see `https://toc.cryptobook.us/`.

Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 229–243. ACM, 2017.

Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://eprint.iacr.org/2000/067`.

Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

Ran Canetti. Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. `https://eprint.iacr.org/2003/239`.

Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369. ACM, 1986.

Daniel Fernandez, Ariel Futoransky, Gustavo Ajzenman, Matias Travizano, and Carlos Sarraute. Wibson protocol for secure data exchange and batch payments. *arXiv:2001.08832*, 2020.

Ariel Futoransky, Emiliano Kargieman, Carlos Sarraute, and Ariel Waissbein. Foundations and applications for secure triggers. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):94–112, 2006.

Ariel Futoransky, Carlos Sarraute, Ariel Waissbein, Daniel Fernandez, Matias Travizano, and Martin Minnoni. Secure exchange of digital goods in a decentralized data marketplace. In *Proceedings of the 2019 Argentine Symposium on Big Data (AGRANDA)*, pages 38–44, 2019.

Ariel Futoransky, Carlos Sarraute, Ariel Waissbein, Matias Travizano, and Daniel Fernandez. WibsonTree: Efficiently preserving seller's privacy in a decentralized data marketplace. *arXiv:2002.03810*, 2020.

Juán Garay, Agelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9057. Springer, Berlin, Heidelberg, 2015.

Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988. ISSN 0097-5397.

Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. Cryptology ePrint Archive, Report 2015/574, 2015. `https://eprint.iacr.org/2015/574`.

Hartwig Mayer, Ismael Bejarano, Daniel Fernandez, Gustavo Ajzenman, Nicolas Ayala, Nahuel Santoalla, Carlos Sarraute, and Ariel Futoransky. BatPay: a gas efficient protocol for the recurrent micropayment of ERC20 tokens. *arXiv:2002.02316*, 2020.

Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 12–19, 2003.

Yusuke Okada, Yoshifumi Manabe, and Tatsuaki Okamoto. An optimistic fair exchange protocol and its security in the universal composability framework. *International Journal of Applied Cryptography*, 1(1):70–77, 2008.

Matias Travizano, Carlos Sarraute, Gustavo Ajzenman, and Martin Minnoni. Wibson: A decentralized data marketplace. In *Proceedings of SIGBPS 2018 Workshop on Blockchain and Smart Contract*, 2018.