Secret Swapping: Two Party Fair Exchange

Alex Dalton[1], David Thomas[2], and Peter Cheung[1]

[1] Imperial College London
{akd19,p.cheung}@ic.ac.uk
[2] University of Southampton
d.b.thomas@southampton.ac.uk

Abstract. Consider two participants who wish to each reveal some secret information to each other, but both will only reveal their secret if there is a guarantee that the other will do so too. Conventionally, whoever sends their secret first makes themselves vulnerable to the possibility that the other party will cheat and won't send their secret in reply. Fair Exchange (FE) protocols are a class of cryptographic construction which allow an exchange like this to occur without either party making themselves vulnerable. It is widely believed that a trusted third party arbitrator is required to intervene in the case of a dispute. We present a FE protocol that allows two parties to exchange secrets, safe in the knowledge that either both secrets will be exchanged, or neither will; without the involvement of a third party. This construction requires that the parties run online interactive processes, with reasonable timeliness requirements on the messages. In this work we provide a scheme for swapping arbitrarily sized secrets with security that reduces to the strength of an underlying symmetric authenticated encryption scheme.

1 Introduction

It's common for two parties to want to trade secrets, but neither party wishes to go first, as to do so would expose themselves to the possibility of the other participant cheating. These types of transactions occur often in the wild: in economic trades where both parties might exchange a cryptographic signature approving the transfer of funds [13]; the output of executed programs could be traded; or a combination of these, a payment for computational work conducted in the cloud. Similarly, corporations or nation state allies may wish to trade sensitive strategic information. Also, some cryptographic protocols end with the assumption that one party will dutifully pass a result to the other. For example, in Multi Party Computation (MPC) constructions[15] the circuit garbler is expected to dutifully reveal the decrypted results of joint calculations to the circuit evaluator.

Currently, this problem is solved by using a third party to mediate the transaction. This third party might be a bank, mutual friend, or some blockchain-backed protocol[9][10][6]. In the naive case the third party holds both secrets until they have been validated, and then forwards them on to each party. Fair Exchange protocols are a class of cryptographic construction improve over the naive case, by reducing the involvement of a third party, and allow for two parties to exchange secrets[1]. Current state-of-the-art Fair Exchange protocols require an offline trusted authority that is only required to intervene in the case of a dispute[2][12]. Reducing the involvement of the third party is one of the key objectives of FE protocol research. It's widely believed that a Fair Exchange construction is not possible without a third party performing arbitration[14].

In this work we present a two party Fair Exchange protocol with security that reduces to a standard assumption: the existence of an IND-CPA secure authenticated encryption scheme.

1.1 Contributions

In this work we:

– Define *Secret Swapping*, a two party Fair Exchange scheme with security relating to the IND-CPA property of an underlying symmetric authenticated encryption scheme.
– Extend standard Fair Exchange definitions to include computational security.
– Prove Secret Swapping has *Computational Completeness* and *Computational Strong Fairness.*

1.2 Assumptions

We assume all parties send a secret which satisfies some property expected by the other party. For example, the property could be that the secret is a valid signature on a predetermined message. This can be enforced with zero knowledge proofs on the ciphertexts as encryptions of expected data. Furthermore, we assume all parties will follow the protocol honestly, up to the moment where a secret is revealed, at which point a malicious party will do what they can to avoid revealing their secret.

We make a non-standard assumption; that there is a "secure" λ and an "insecure" λ' such that the adversarial advantage against each is separated by a factor M, and M is independent of λ'.

Assumption 1 *There exists security parameter λ, λ', and M such that*

$$M\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A}}(\lambda') \tag{1}$$

where M is independent of λ' and λ is considered secure, but λ' is considered bruteforcible.

Consider an instantiation of a cryptosystem deemed "insecure" $\mathcal{E}(\lambda')$ with a key space bruteforcible within one month. There exists a corresponding instantiation $\mathcal{E}(\lambda)$ with key space bruteforcible in 1000 years. In this case $M \approx 10,000$. If λ and λ' need to be updated because of algorithmic advances or hardware improvements M is unaffected.

M is in fact a ratio between the time taken to attack security parameter λ and security parameter λ'. In conventional security practice the security parameter is chosen to satisfy some idea of the time taken to attack the system. As attack methods are developed, λ is adaptively increased. It is easy to see that M is therefore not directly related to λ or λ' and is instead dependent on the relationship between the two. Therefore however M is chosen, it is constant as λ and λ' adaptively grow.

As a concrete example, consider two instantiation of AES and an adversary with access to common commodity computer hardware. Suppose we select a "secure" instantiation of AES bruteforcible in 1000 years. Obviously individual ideas of security vary from setting to setting. But in FE the goal is for targets to eventually exchange secrets, so a key space bruteforcible in 1000 years on commodity hardware could very well be considered secure. With some back of the envelope calculations this would result in a key length of $\lambda \approx 84$ bits. The complimentary "bruteforcible" instantiation, with key space bruteforcible in a month, would have a key length of $\lambda' \approx 58$ bits. In this case we have $M \approx 2^{26}$. Note that as we adaptively change λ and λ' to stay ahead of algorithmic improvements M will stay fixed.

We also make the standard assumption that there exists a IND-CPA secure symmetric authenticated encryption scheme.

1.3 Notation

Functions are denoted with a specific $\mathsf{Function}$ font. $\leftarrow$ indicates assignment. We make use of standard cryptographic notation: k is an cryptographic key, symmetric encryption schemes are defined as a set of functions $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ and λ is a security parameter. Functions that grow negligibly in the security parameter are denoted with $\mathsf{negl}(\lambda)$. At times we make use of vectors to indicate an ordered set of data, these are denoted by a variable in bold $\mathbf{v}$. The ith element of the vector $\mathbf{v}$ is denoted by $\mathbf{v}[i]$. $[x; n]$ is used as shorthand for an n length vector with the value x at every position.

We make black-box use of oblivious transfer[4]. Oblivious transfer allows a sender to offer up a vector of data, $\mathbf{y}$, and a receiver is able to access a single element indexed at i without the sender learning which element the receiver accessed, and without the receiver learning any of the unaccessed data in the vector $\mathbf{y}$. The use of oblivious transfer is noted with $x \xleftarrow{OT} \mathbf{y}[i]$, where x is the local variable of the recipient, $\mathbf{y}$ is an ordered vector of data stored on the sender, and i is the index in $\mathbf{y}$ accessed as part of the oblivious transfer.

The two participants in our secret swapping scheme are referenced as party X and party Y. Variables associated with either party are usually subscripted with the appropriate letter. In the protocol description "X :" is a prefix used to indicate the operation is executed by party X.

2 Background

Many references to underlying cryptographic building blocks are kept intentionally abstract. In this section we define the properties we make use of later in this work.

2.1 Fair Exchange

These definitions are adapted from [12]. In a Fair Exchange protocol party X holds one piece of data, d_X, while party Y holds a different piece of data, d_Y. The protocol is *Complete* if, with honest parties, X learns d_Y and Y learns d_X.

Definition 1 (Complete). *With honest parties X and Y, and respective data d_X and d_Y, when the protocol terminates X has learned d_Y and Y has learned d_X.*

The difficulty arises from constructing a protocol in which a dishonest participant is not able to discover the secret of the other participant without giving up their own secret. There should be no way of X learning anything about d_Y without offering up d_X to Y. The same is true for Y in relation to d_X. This property is *Strong Fairness.*

Definition 2 (Strong Fairness). *When the protocol has completed, either X has d_Y or Y has gain no information about d_X. Similarly, when the protocol has completed either Y has learned d_X or X has gained no information about d_Y.*

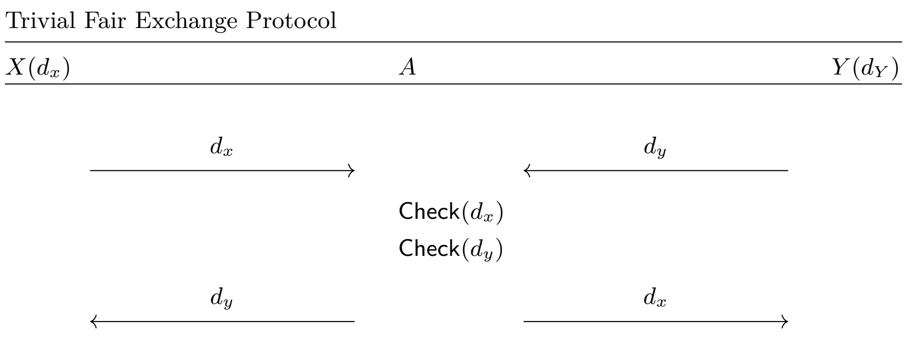


Fig. 1: Trivial Fair Exchange protocol with a trusted third party arbitrator, A.

A naive solution to the problem is to allow all messages to pass through a trusted third party, A. This approach is shown in Figure 1. The current state-of-the-art only requires the trusted third party to intervene in the case of a dispute[12].

It's widely believed that strong fairness is impossible without the use of a trusted third party[14]. This belief stems from a result in which two party Fair Exchange protocols can be used to construct consensus schemes, and as there are no deterministic algorithms for consensus[8], there can be no deterministic two-party Fair Exchange protocols. This result required that no timing restrictions are placed on the messages between parties. A close reading of this work poses the question; can a non-deterministic two-party Fair Exchange protocol be constructed with *Strong Fairness* if reasonable timeliness restrictions are placed on the messages?

2.2 Symmetric IND-CPA Secure Authenticated Encryption Schemes

A symmetric encryption scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is an authenticated encryption scheme if, along with confidentiality of data, it provides integrity and authenticity. The authenticated encryption scheme is defined by the following functions:

- $\mathsf{KGen}(\lambda) \rightarrow \mathsf{k}$; generate key k.
- $\mathsf{Enc}_\mathsf{k}(m) \rightarrow c$; encrypt message m to ciphertext c.
- $\mathsf{Dec}_\mathsf{k}(c) \rightarrow m$ or $\perp$; decrypt ciphertext c to message m, with an invalid combination of c and k decrypting to $\perp$.

In the case a ciphertext has been tampered with, or does not originate from the expected sender Dec outputs $\perp$. Authenticated encryption schemes are usually composed from encryption schemes and message authentication codes.

```
$\mathsf{INDCPAGame}_1(\lambda)$
1 : $\mathsf{k} \leftarrow \mathsf{KGen}(\lambda), b \leftarrow\!\$ \{0,1\}$
2 : $m_0, m_1 \leftarrow \mathcal{A}^{\mathsf{OEnc}_\mathsf{k}}()$
3 : $c \leftarrow \mathsf{Enc}_\mathsf{k}(m_b)$
4 : $b' \leftarrow \mathcal{A}^{\mathsf{OEnc}_\mathsf{k}}(c)$
5 : **if** $b = b'$ **then return** $1$
6 : **else return** $0$
```

Fig. 2: IND-CPA security game. A key is generated, k as well as random bit b. There is an implicit adversary state maintained between the two calls to the adversary $\mathcal{A}$. The adversary wins if they can differentiate an encryption of one of two messages of their choice. They are given access to an encryption oracle OEnc_k which is able to provide valid encryptions interactively to the adversary.

A scheme is IND-CPA secure if it is indistinguishable under a chosen plaintext attack[3]. Consider the security game detailed in Figure 2 and the following definition.

Definition 3 (Symmetric IND-CPA **Secure Authenticated Encryption Scheme).** *Consider symmetric authenticated encryption scheme* $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*. The advantage of an adversary* $\mathcal{A}$ *is the* IND-CCA *security game, detailed in Figure 2, as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{ind-cpa}}(\lambda) = 2\Pr[\mathsf{INDCPAGame}(\lambda) = 1] - 1 \tag{2}$$

A cryptosystem $\mathcal{E}$ *is* IND-CPA *secure if for any probabilistic polynomial time adversary* $\mathcal{A}$

$$\mathsf{Adv}_{\mathcal{A}}^{\text{ind-cpa}}(\lambda) \leq \mathsf{negl}(\lambda) \tag{3}$$

2.3 Oblivious Transfer

An oblivious transfer protocol allows one person to send another a single piece of information from a set of information, without the Sender knowing which was element from the set was sent, and without the Receiver learning any of the other pieces of information from the set. There are existing constructions for fully generalised k-in-n oblivious transfer[4]. We make black box use of oblivious transfer in our construction.

3 Extended Definitions for Fair Exchange

In this section we extend known FE definitions to be more suitable for the work presented in this paper. The extensions adapt existing FE definitions from the domain of perfect security to computational security. To extend the Fair Exchange notion of *Completeness* we provide a definition for *Computational Completeness.*

Definition 4 (Computational Completeness). *Given* X *with data* d_X *and* Y *with data* d_Y*. If both* X *and* Y *act honestly,* X *learns* d_Y *and* Y *learns* d_X *with probability*

$$\Pr[X(d_Y) \wedge Y(d_X)] \approx 1 \tag{4}$$

Similarly, we provide a definition for *Computational Strong Fairness.*

Definition 5 (Computational Strong Fairness). *For arbitrary parties* P *and* Q *define the advantage in the fairness game of adversarial party* P *as*

$$\mathsf{Adv}_{\mathcal{A}_P}^{\text{fairness}}(\lambda) = \Pr[\mathsf{FairnessGame}(\lambda) = 1] \tag{5}$$

Consider protocol transcript δ*. For a given* δ*, if the ability of party* Y *to extract* d_X *from* δ *is computationally intractable, then for any probabilistic polynomial time adversary* $\mathcal{A}_X$

$$\mathsf{Adv}_{\mathcal{A}_X}^{\text{fairness}}(\lambda) \leq \mathsf{negl}(\lambda) \tag{6}$$

Similarly, if the ability of X *to extract* d_Y *from* δ *is computationally intractable, for any probabilistic polynomial time adversary* $\mathcal{A}_Y$

$$\mathsf{Adv}_{\mathcal{A}_Y}^{\text{fairness}}(\lambda) \leq \mathsf{negl}(\lambda) \tag{7}$$

```
FairnessGame_1(λ)
1 : $\delta \leftarrow\!\$\, \Delta$
2 : $d'_Q \leftarrow \mathcal{A}_P(\delta)$
3 : if $d_Q = d'_Q$ then return $1$
4 : else return $0$
```

Fig. 3: Fairness security game for a Fair Exchange protocol. For a complete or partial protocol transcript δ randomly sampled from the space of possible transcripts Δ, and adversary acting as party P given by $\mathcal{A}_P$. If the adversary is able to extract the data of the other participant, d_Q, they win. We use P and Q to indicate the different parties and separate them from the specific roles X and Y have in our scheme.

4 Secret Swapping

The intuition behind the secret Swapping Scheme is that we can produce a protocol in which there is a single success state in which both parties learn about each other's secrets, and many failure states. This way if a party terminates early when they successfully receive the other party's secret, the spurned party can infer some information to help derive their secret.

4.1 Description

Both parties negotiate a λ and λ' such that λ is not reasonably bruteforcible, perhaps with an expected time-to-bruteforce of 1000 years, but λ' is considered bruteforcible in some long, but reasonable, time-frame; say one month. We denote the expected time to attack $\mathcal{E}(\lambda)$ as T, and the expected time to bruteforce $\mathcal{E}(\lambda')$ as T'.

$$T' \ll T \tag{8}$$

Messages in the protocol are the expected to have a timeliness of t such that

$$t \ll T' \tag{9}$$

With the above example time frames t could be set to one hour. Messages that do not maintain this timeliness cause the protocol to fail. Both parties then blacklist the exchange, and do not allow it to be restarted.

Given λ and λ' both parties then derive, derive value M, such that

$$M\mathsf{Adv}_{\mathcal{A}}^{\text{ind-cpa}}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\text{ind-cpa}}(\lambda') \tag{10}$$

Another way of expressing M is as the ratio between the time expected to bruteforce both instantiation of $\mathcal{E}$.

$$M = \frac{T}{T'} \tag{11}$$

X and Y generate two keys each for a symmetric IND-CPA authenticated encryption scheme $\mathcal{E}$.

$$X : \mathsf{k}_{X0} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda) \qquad Y : \mathsf{k}_{Y0} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda) \tag{12}$$
$$X : \mathsf{k}_{X1} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda) \qquad Y : \mathsf{k}_{Y1} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda) \tag{13}$$

They then select one of the keys at random, and encrypt their data under this key. As both keys are indistinguishable, we'll assume both parties selected key 0.

$$X : c_X \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{k}_{X0}}(d_X) \qquad Y : c_Y \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{k}_{Y0}}(d_Y) \tag{14}$$

Both parties then freely exchange the ciphertexts. Party Y then generates a blinding value b. b is a bit-vector of size λ'. They use this value to blind the least significant bits of the key they used.

$$Y : b \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda') \tag{15}$$
$$Y : \mathsf{k}'_{Y0} \leftarrow \mathsf{k}_{Y0} \oplus b \tag{16}$$

Both parties randomly sample m such that $0 \leq m \leq M$ and construct large M-sized vector $\mathbf{k}$, where k_1 is in every position except index m. k_0 is at index m in the vector $\mathbf{k}$. Party Y uses k'_{Y0} instead of k_{Y0}.

$$X : m_X \leftarrow\!\!\$\; \{0, \ldots, M\} \qquad Y : m_Y \leftarrow\!\!\$\; \{0, \ldots, M\} \tag{17}$$
$$X : \mathbf{k}_X \leftarrow [\mathsf{k}_{X1}; M] \qquad Y : \mathbf{k}_Y \leftarrow [\mathsf{k}_{Y1}; M] \tag{18}$$
$$X : \mathbf{k}_X[m_X] \leftarrow \mathsf{k}_{X0} \qquad Y : \mathbf{k}_Y[m_Y] \leftarrow \mathsf{k}'_{Y0} \tag{19}$$

X then uses oblivious transfer to learn the element at position m_X in Y's vector $\mathbf{k}_Y$.

$$X : \mathsf{k}_Y \xleftarrow{OT} \mathbf{k}_Y[m_X] \tag{20}$$

X can bruteforce the key they received, but not in a timely manner. Nor do they know if they managed to hit the correct element in Y's vector. They must continue the protocol because of the timeliness restriction.

Y then uses oblivious transfer to learn the element at position m_Y in X's vector $\mathbf{k}_X$

$$Y : \mathsf{k}_X \xleftarrow{OT} \mathbf{k}_X[m_Y] \tag{21}$$

Y can use the key they received to attempt to decrypt c_X.

$$Y : d_X \text{ or } \perp \leftarrow \mathcal{E}.\mathsf{Dec}(c_X) \tag{22}$$

If Y succeeds they can let X know, and release b to X in good faith. Otherwise the protocol failed and both parties need to restart. Y can prove the failure to X by revealing their value for k_X.

This protocol is shown diagrammatically in Figure 4.

Secret Swapping Protocol

$X(\mathcal{E}, \lambda, d_X, B, M)$		$Y(\mathcal{E}, \lambda, d_Y, B, M)$
$\mathsf{k}_{X0} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda)$		$\mathsf{k}_{Y0} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda)$
$\mathsf{k}_{X1} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda)$		$\mathsf{k}_{Y1} \leftarrow \mathcal{E}.\mathsf{KGen}(\lambda)$
$c_X \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{k}_{X0}}(d_X)$		$c_Y \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{k}_{Y0}}(d_Y)$
	$\xleftarrow{\quad c_Y \quad}$	
	$\xrightarrow{\quad c_X \quad}$	
		$b \leftarrow\!\!\$\ \{0, \ldots, B\}$
		$\mathsf{k}'_{Y0} \leftarrow \mathsf{k}_{Y0} \oplus b$
$m_X \leftarrow\!\!\$\ \{0, \ldots, M\}$		$m_Y \leftarrow\!\!\$\ \{0, \ldots, M\}$
$\mathbf{k}_X \leftarrow [\mathsf{k}_{X1}; M]$		$\mathbf{k}_Y \leftarrow [\mathsf{k}_{Y1}; M]$
$\mathbf{k}_X[m_X] \leftarrow \mathsf{k}_{X0}$		$\mathbf{k}_Y[m_Y] \leftarrow \mathsf{k}'_{Y0}$
	$\xleftarrow{\ \mathsf{k}'_Y \overset{OT}{\longleftarrow} \mathbf{k}_Y[m_X]\ }$	
	$\xrightarrow{\ \mathsf{k}_X \overset{OT}{\longleftarrow} \mathbf{k}_X[m_Y]\ }$	
		$o \leftarrow \mathcal{E}.\mathsf{Dec}_{\mathsf{k}_X}(c_X)$
		if $o = \perp$ **then** $r \leftarrow \mathsf{k}_X$
		else $r \leftarrow b$
	$\xleftarrow{\quad r \quad}$	
if $r = \mathsf{k}_{X1}$ **then return** $\perp$		**if** $o = \perp$ **then return** $\perp$
else return $\mathcal{E}.\mathsf{Dec}_{\mathsf{k}'_Y \oplus r}(c_Y)$		**else return** o

Fig. 4: Secret swapping protocol overview.

4.2 Proofs

Theorem 1 (Computational Completeness). *For honest X and Y the probability of successfully swapping their secrets is*

$$\Pr[X(d_Y) \wedge Y(d_X)] \approx 1 \tag{23}$$

Proof. On a single run of the secret swapping protocol, X and Y successfully exchanging data if $m_X = m_Y$. This value is the index at which X and Y store the keys they used in the vector $\mathbf{k}$ and it is also the element they access via oblivious transform from the other party's vector $\mathbf{k}$.

m is a value between 0 and M. Therefore the probability of exchanging data is on a single run of the protocol is:

$$\Pr[X(d_Y) \wedge Y(d_X)] = \frac{1}{M} \tag{24}$$

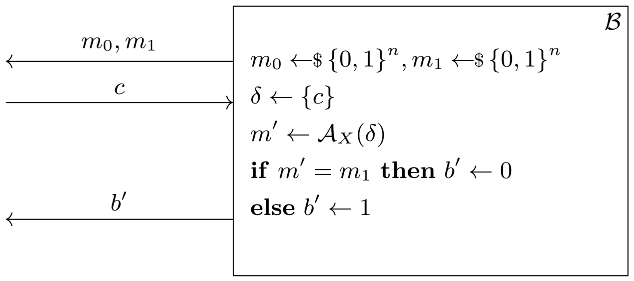
By Assumption 1, M is a constant. It follows that the protocol can be reexecute as many times as required for the probability of a successful data exchange to be arbitrarily close to 1.

$$\therefore \Pr[X(d_Y) \wedge Y(d_X)] \approx 1 \tag{25}$$

The following Lemmas prove qualities about the fairness of the system after each message is sent. In each subsequent Lemma the protocol transcript δ is extended with the next message, and the adversary $\mathcal{A}$ takes on the role of the recipient of the latest message, party X or Y.

Lemma 1. *When $\delta = \{c_Y\}$*

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A}}(\lambda) \tag{26}$$



```
$m_0 \leftarrow\$ \{0,1\}^n, m_1 \leftarrow\$ \{0,1\}^n$
$\delta \leftarrow \{c\}$
$m' \leftarrow \mathcal{A}_X(\delta)$
**if** $m' = m_1$ **then** $b' \leftarrow 0$
**else** $b' \leftarrow 1$
```

Fig. 5: Adversary $\mathcal{B}$; capable of winning the IND-CPA security game given the existence of adversary $\mathcal{A}_X$.

Proof. Consider adversary $\mathcal{A}_X$ capable of extracting d_Y from δ. This could easily be used to construct $\mathcal{B}$, in Figure 5, such that it forwards ciphertexts to $\mathcal{A}_X$ to provide a decryption.

$$\therefore \mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) \leq \mathsf{Adv}^{\text{ind}-\text{cpa}}_{\mathcal{A}}(\lambda) \tag{27}$$

Lemma 2. *When* $\delta = \{c_Y, c_X\}$

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_Y}(\lambda) \leq \mathsf{Adv}^{\text{ind}-\text{cpa}}_{\mathcal{A}}(\lambda) \tag{28}$$

This proof is omitted for brevity. It follows the same outline as the proof for Lemma 1 but with party Y and X swapped, and $\mathcal{B}$ randomly samples a placeholder c_Y in the construction of the transcript δ.

Lemma 3. *When* $\delta = \left\{c_Y, c_X, \mathsf{k}'_Y\right\}$

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) = \mathsf{Adv}^{\text{ind}-\text{cpa}}_{\mathcal{A}}(\lambda) \tag{29}$$

Proof. There are two possible situations; the first one is that k'_Y is a blinded version of the key used to encrypt ciphertext c_Y, the other is that they are unrelated. In the latter case this proof is trivial.

The former case occurs with probability $\frac{1}{M}$. This relationship is reflected in the security parameter λ' which is defined as the security parameter of an instantiation of $\mathcal{E}$ which is bruteforcible in a long but reasonable time-frame.

The advantage of $\mathcal{A}_X$ in the $\mathsf{FairnessGame}$ is therefore

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) = \frac{1}{M}\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{B}}(\lambda') \tag{30}$$

for some adversary $\mathcal{B}$ which can make calls to $\mathcal{A}_X$.

The definition of the constant M in the protocol is given by

$$M\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{B}}(\lambda) = \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{B}}(\lambda') \tag{31}$$

It follows that the advantage of $\mathcal{A}_X$ in the $\mathsf{FairnessGame}$ is

$$\therefore \mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) = \frac{1}{M}\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{B}}(\lambda') \tag{32}$$

$$= \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{B}}(\lambda) \tag{33}$$

Lemma 4. *When* $\delta = \left\{c_Y, c_X, \mathsf{k}'_Y, \mathsf{k}'_X\right\}$ *and, given a long but reasonable amount of time,* X *is unable to mount a bruteforce attack against* c_Y

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_Y}(\lambda) \leq \mathsf{Adv}^{\text{ind}-\text{cpa}}_{\mathcal{A}}(\lambda) \tag{34}$$

Proof. As, given k'_Y, X is unable to mount a bruteforce attack against c_Y in any reasonable amount of time, it must be the case that $m_X \neq m_Y$. Because of this, k'_X is completely unrelated to c_X and so is no help for decrypting it. The rest of the proof follows from Lemma 2.

Theorem 2 (Computational Strong Fairness). *Given $\mathcal{E}$, an* IND-CPA *secure authenticated encryption scheme. Where $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. In the Secret Swapping protocol, if the ability of Y to extract d_X from transcript δ is computationally intractable, then for any probabilistic polynomial time adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_Y}(\lambda) \leq \mathsf{negl}(\lambda) \tag{35}$$

Similarly, if the ability of X to extract d_Y from transcript δ is computationally intractable, then for any probabilistic polynomial time adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) \leq \mathsf{negl}(\lambda) \tag{36}$$

Proof. In the first case, when party Y is acting maliciously, the transcript must be either $\delta = \{c_Y, c_X\}$ or $\delta = \left\{c_Y, c_X, \mathsf{k}'_Y, \mathsf{k}'_X\right\}$. The former case is proved in Lemma 2 and the later is proved in Lemma 4. It follows that for all possible situations, if the ability of X to extract d_Y from δ is intractable then

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_Y}(\lambda) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A}}(\lambda) \tag{37}$$

$$\implies \mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_Y}(\lambda) \leq \mathsf{negl}(\lambda) \tag{38}$$

The second case, when party X is acting maliciously, is slightly more complex. Consider the timeliness restriction on the messages from X as part of the protocol definition. This restriction is much shorter than the expected time to bruteforce $\mathcal{E}$ with security parameter λ'. The timeliness requirement hardens the protocol against the ability of $\mathcal{A}_X$ to preemptively "test" bruteforce the case when $\delta = \left\{c_Y, c_X, \mathsf{k}'_Y\right\}$. Without this restriction the result from Lemma 3 would not apply, as the $\frac{1}{M}$ factor would no longer be relevant in Equation 30, as the bruteforce attack could be attempted, and on a failure X could pretend to be honest and continue through repeat protocol executions until the bruteforce attack works. As it is, with a reasonable timeliness restriction on messages this is not possible.

Given this restriction Lemmas 1 and 3 prove the advantage of $\mathcal{A}_X$ must be

$$\mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A}}(\lambda) \tag{39}$$

$$\implies \mathsf{Adv}^{\text{fairness}}_{\mathcal{A}_X}(\lambda) \leq \mathsf{negl}(\lambda) \tag{40}$$

4.3 Evaluation

We have proved that Secret Swapping has both the *Computational Completeness* and *Computational Strong Fairness* properties without the need for a trusted third party arbitrator.

A glaring deficiency in our scheme is that as λ is made arbitrarily large, M must grow exponentially to reflect the difference between λ and λ'. With this, the probability of a successful data transfer per execution of the protocol becomes vanishingly small. λ is therefore restricted to smaller values that are deemed "secure enough".

The massive failure rate of our scheme, $\frac{M-1}{M}$ is mitigated by the fact M is a constant, and so the protocol is expected to be run a constant M times before a success. However, the multiple runs of the protocol cannot be parallelised, they must be executed sequentially. If they were parallelised Lemma 3 would be false.

5 Related Work

All previous Fair Exchange protocols use a trusted third party in some way. The previous state-of-the-art only required a third party to mediate in the case of a dispute[12].

There is some notion of a naive two party Fair Exchange wherein X and Y exchange ciphertexts and then alternately release individual bits of the secret key. This obviously has some unfortunate qualities, namely that X and Y need to be able to leverage a similar amount of computational power. If X terminates early because the remaining key bits are bruteforcible, the remaining bits for Y should also be bruteforcible.

There is a body of work solving the problem with financial incentives to operate honestly[11]. In such work a misbehaving party must forfeit previously posted financial assets. The advent of blockchain has allowed a decentralised ledger to stand in for a trusted third party[9][10][6][7]. Etherium smart contracts can encode the behaviour of a trusted third party, and release Etherium tokens should certain conditions be met[5]. This is unfortunately very expensive to implement, relies on the security of a blockchain and the honesty of it's nodes, and is restricted to the exchange of cryptocurrency tokens.

6 Conclusion

To the best of our knowledge this is the first Fair Exchange protocol not require any third party involvement. This result was previously assumed to be impossible, but in fact it is only impossible for a specific configuration of the problem. A widely cited impossibility result showed that no deterministic two party Fair Exchange protocol existed without any timeliness restrictions on the messages[14]. We demonstrate that a non-deterministic two party Fair Exchange protocol can exist provided there are sensible timeliness restrictions messages.

Our protocol needs to be repeated a constant, but very large, number of times in order for it to terminate successfully. It may therefore never be considered practical to implement.

7 Further Work

There is clear value in two party Fair Exchange, which our construction shows is possible. However, as highlighted in the Evaluation, this protocol is not without flaws. The massive failure rate of $\frac{M-1}{M}$ requires the protocol to be run an expected M times. Depending on how cautiously M is selected and the security parameter of the "secure" encryption scheme λ this can be very large. The bandwidth requirements can be very large.

The security of the scheme is forced to stay in the realm of "good enough" as to keep M relatively small. There is potential for further work in hardening the scheme in this regard.

References

1. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: Proceedings of the 4th ACM Conference on Computer and Communications Security. p. 7–17. CCS '97, Association for Computing Machinery, New York, NY, USA (1997). https://doi.org/10.1145/266420.266426, https://doi.org/10.1145/266420.266426
2. Bao, F., Deng, R., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No.98CB36186). pp. 77–85 (1998). https://doi.org/10.1109/SECPRI.1998.674825
3. Bellare, M., Rogaway, P.: Introduction to modern cryptography (2005)
4. Brassard, G., Crepeau, C., Robert, J.M.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 234–238. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
5. Buterin, V.: A next generation smart contract & decentralized application platform (2013)
6. Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L.: Zero-knowledge contingent payments revisited: Attacks and payments for services. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 229–243. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134060, https://doi.org/10.1145/3133956.3134060
7. Eckey, L., Faust, S., Schlosser, B.: Optiswap: Fast optimistic fair exchange. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. p. 543–557. ASIA CCS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3320269.3384749, https://doi.org/10.1145/3320269.3384749
8. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (apr 1985). https://doi.org/10.1145/3149.214121, https://doi.org/10.1145/3149.214121
9. Goldfeder, S., Bonneau, J., Gennaro, R., Narayanan, A.: Escrow protocols for cryptocurrencies: How to buy physical goods using bitcoin. In: Kiayias, A. (ed.) Financial Cryptography and Data Security. pp. 321–339. Springer International Publishing, Cham (2017)
10. Jakobsson, M.: Ripping coins for a fair exchange. In: Guillou, L.C., Quisquater, J.J. (eds.) Advances in Cryptology — EUROCRYPT '95. pp. 220–230. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
11. Janin, S., Qin, K., Mamageishvili, A., Gervais, A.: Filebounty: Fair data exchange. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). pp. 357–366 (2020). https://doi.org/10.1109/EuroSPW51379.2020.00056
12. Küpçü, A., Lysyanskaya, A.: Usable optimistic fair exchange. Computer Networks **56**(1), 50–63 (2012). https://doi.org/https://doi.org/10.1016/j.comnet.2011.08.005, https://www.sciencedirect.com/science/article/pii/S138912861100301X
13. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
14. Pagnia, H., Darmstadt, F.C.G.: On the impossibility of fair exchange without a trusted third party (1999)
15. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982). pp. 160–164 (1982). https://doi.org/10.1109/SFCS.1982.38