Optimistic Fair Priced Oblivious Transfer

A. Rial B. Preneel

Katholieke Universiteit Leuven - ESAT-COSIC
IBBT

Africacrypt 2010

Priced Oblivious Transfer: Definition

Vendor

$(m_1, \ldots, m_N), (p_1, \ldots, p_N)$

$\longleftarrow$

$\longrightarrow$

Buyer

$\tau \in \{1, \ldots, N\}$

m_τ

Properties:

- $\mathcal{V}$ does not learn τ.
- $\mathcal{B}$ does not get any information about other messages.
- $\mathcal{B}$ pays price p_τ.

Priced Oblivious Transfer: Definition

Vendor

$(m_1, \ldots, m_N), (p_1, \ldots, p_N)$

Buyer

$\tau \in \{1, \ldots, N\}$

$\longleftarrow$

$\longrightarrow$

m_τ

Properties:

- $\mathcal{V}$ does not learn τ.
- $\mathcal{B}$ does not get any information about other messages.
- $\mathcal{B}$ pays price p_τ.

Priced Oblivious Transfer: Definition

Vendor

$(m_1, \ldots, m_N), (p_1, \ldots, p_N)$

Buyer

$\tau \in \{1, \ldots, N\}$

m_τ

Properties:

- $\mathcal{V}$ does not learn τ.
- $\mathcal{B}$ does not get any information about other messages.
- $\mathcal{B}$ pays price p_τ.

Priced Oblivious Transfer: Definition

Vendor		Buyer
$(m_1, \ldots, m_N), (p_1, \ldots, p_N)$	$\longleftarrow$	$\tau \in \{1, \ldots, N\}$
	$\longrightarrow$	m_τ

Properties:

- $\mathcal{V}$ does not learn τ.
- $\mathcal{B}$ does not get any information about other messages.
- $\mathcal{B}$ pays price p_τ.

Priced Oblivious Transfer: Construction

Vendor

Buyer

ac_0 ← ac_0

Prepaid Mechanism

- $\mathcal{B}$ makes an initial deposit to $\mathcal{V}$.
- At each purchase, the price is debited from the deposit.
- $\mathcal{V}$ learns neither the price nor the deposit.

Priced Oblivious Transfer: Construction

Vendor

Buyer

ac_0

ac_0

Prepaid Mechanism

- $\mathcal{B}$ makes an initial deposit to $\mathcal{V}$.
- At each purchase, the price is debited from the deposit.
- $\mathcal{V}$ learns neither the price nor the deposit.

Priced Oblivious Transfer: Construction

Vendor

Buyer

ac_0

$\longleftarrow ac_0$

Prepaid Mechanism

- $\mathcal{B}$ makes an initial deposit to $\mathcal{V}$.
- At each purchase, the price is debited from the deposit.
- $\mathcal{V}$ learns neither the price nor the deposit.

Priced Oblivious Transfer: Construction

Vendor

Buyer

ac_0

$\leftarrow ac_0$

Prepaid Mechanism

- $\mathcal{B}$ makes an initial deposit to $\mathcal{V}$.
- At each purchase, the price is debited from the deposit.
- $\mathcal{V}$ learns neither the price nor the deposit.

Priced Oblivious Transfer: Security

Previous Work

Half-Simulation secure schemes [AIR01, Tob03].

- Vulnerable under attack in [DNO08].

UC-secure scheme [RKP09].

- Inefficient.

Efficient Full-Simulation Secure POT

Priced Oblivious Transfer: Security

Previous Work

Half-Simulation secure schemes [AIR01, Tob03].

- Vulnerable under attack in [DNO08].

UC-secure scheme [RKP09].

- Inefficient.

Efficient Full-Simulation Secure POT

Priced Oblivious Transfer: Security

Previous Work

Half-Simulation secure schemes [AIR01, Tob03].

- Vulnerable under attack in [DNO08].

UC-secure scheme [RKP09].

- Inefficient.

Efficient Full-Simulation Secure POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Priced Oblivious Transfer: Fairness

Previous Work

Usually, e-commerce protocols are analyzed to prove their fairness [Kre04].

- Non privacy-preserving protocols [EGL85, Gol83].
- Privacy-preserving protocols that provide buyers' anonymity [RR01].

However, no fair POT scheme has been proposed.

- Malicious $\mathcal{V}$ can claim $\mathcal{B}$ ran out of funds.
- Malicious $\mathcal{V}$ can deny delivery.
- Malicious $\mathcal{B}$ can falsely accuse an honest $\mathcal{V}$.

Fair POT

Outline

1. Efficient Priced Oblivious Transfer
   - Construction
   - Comparison with Previous Work
2. Optimistic Fair POT
   - Definition
   - Construction

Outline

Outline

Overview

Our POT scheme is based on the OT scheme of [CNS07] and thus follows an assisted decryption approach.

Generic POT scheme

$\mathcal{V}((m_i, p_i)_{i=1}^{N})$	**Initialization**	$\mathcal{B}(ac_0)$
$sk, \forall i, C_i = \mathrm{POTInitV}(m_i, p_i)$	$(C_1, \ldots, C_N)$ →	Verify $(C_1, \ldots, C_N)$
ac_0	← (ac_0)	ac_0
$\mathcal{V}$	**Transfer**	$\mathcal{B}(\tau)$
$\mathrm{POTVerReq}(Q)$	← (Q)	$(Q, Qpriv) = \mathrm{POTReq}(C_\tau)$
$R = \mathrm{POTResp}(Q, sk)$	(R) →	$\mathrm{POTVerResp}(R)$
		$(m_\tau) = \mathrm{POTComplete}(R, Qpriv)$

Overview

Our POT scheme is based on the OT scheme of [CNS07] and thus follows an assisted decryption approach.

Generic POT scheme

$\mathcal{V}((m_i, p_i)_{i=1}^N)$	**Initialization**	$\mathcal{B}(ac_0)$
$sk, \forall i, C_i = \text{POTInitV}(m_i, p_i)$	$(C_1, \ldots, C_N)$ →	Verify $(C_1, \ldots, C_N)$
ac_0	← (ac_0)	ac_0
$\mathcal{V}$	**Transfer**	$\mathcal{B}(\tau)$
$\text{POTVerReq}(Q)$	← (Q)	$(Q, Qpriv) = \text{POTReq}(C_\tau)$
$R = \text{POTResp}(Q, sk)$	(R) →	$\text{POTVerResp}(R)$
		$(m_\tau) = \text{POTComplete}(R, Qpriv)$

Details: Initialization

$\mathcal{V}$ computes the ciphertexts $(C_1, \ldots, C_N)$.

- Computes bilinear map setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.
- Pick secret key $h \in \mathbb{G}$.
- Ciphertext $C_i = (A_i = g^{1/(x+p_i)}, B_i = e(h, A_i) \cdot m_i, p_i)$.

$\mathcal{B}$ verifies each A_i and makes the initial deposit ac_0.

Details: Initialization

$\mathcal{V}$ computes the ciphertexts $(C_1, \ldots, C_N)$.

- Computes bilinear map setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.
- Pick secret key $h \in \mathbb{G}$.
- Ciphertext $C_i = (A_i = g^{1/(x+p_i)}, B_i = e(h, A_i) \cdot m_i, p_i)$.

$\mathcal{B}$ verifies each A_i and makes the initial deposit ac_0.

Details: Initialization

$\mathcal{V}$ computes the ciphertexts $(C_1, \ldots, C_N)$.

- Computes bilinear map setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.
- Pick secret key $h \in \mathbb{G}$.
- Ciphertext $C_i = (A_i = g^{1/(x+p_i)}, B_i = e(h, A_i) \cdot m_i, p_i)$.

$\mathcal{B}$ verifies each A_i and makes the initial deposit ac_0.

Details: Initialization

$\mathcal{V}$ computes the ciphertexts $(C_1, \ldots, C_N)$.

- Computes bilinear map setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.
- Pick secret key $h \in \mathbb{G}$.
- Ciphertext $C_i = (A_i = g^{1/(x+p_i)}, B_i = e(h, A_i) \cdot m_i, p_i)$.

$\mathcal{B}$ verifies each A_i and makes the initial deposit ac_0.

Details: Initialization

$\mathcal{V}$ computes the ciphertexts $(C_1, \ldots, C_N)$.

- Computes bilinear map setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.
- Pick secret key $h \in \mathbb{G}$.
- Ciphertext $C_i = (A_i = g^{1/(x+p_i)}, B_i = e(h, A_i) \cdot m_i, p_i)$.

$\mathcal{B}$ verifies each A_i and makes the initial deposit ac_0.

Transfer phase "j": Request

$\mathcal{B}$ computes a request (POTReq) for item τ:

- $\mathcal{B}$ picks $v \leftarrow \mathbb{Z}_p$ and blinds $V = A_\tau^v$, computes a commitment C_j to new deposit value $ac_{j-1} - p_\tau$ and a proof that:
  - She possesses a signature on price p_τ.
  - C_j commits to $ac_{j-1} - p_\tau$.
  - C_j commits to a non-negative value [CCS08].

Transfer phase “j”: Request

$\mathcal{B}$ computes a request (POTReq) for item τ:

- $\mathcal{B}$ picks $v \leftarrow \mathbb{Z}_p$ and blinds $V = A_\tau^v$, computes a commitment C_j to new deposit value $ac_{j-1} - p_\tau$ and a proof that:
  - She possesses a signature on price p_τ.
  - C_j commits to $ac_{j-1} - p_\tau$.
  - C_j commits to a non-negative value [CCS08].

Transfer phase "j": Request

$\mathcal{B}$ computes a request (POTReq) for item τ:

- $\mathcal{B}$ picks $v \leftarrow \mathbb{Z}_p$ and blinds $V = A_\tau^v$, computes a commitment C_j to new deposit value $ac_{j-1} - p_\tau$ and a proof that:
  - She possesses a signature on price p_τ.
  - C_j commits to $ac_{j-1} - p_\tau$.
  - C_j commits to a non-negative value [CCS08].

Transfer phase “j”: Request

$\mathcal{B}$ computes a request (POTReq) for item τ:

- $\mathcal{B}$ picks $v \leftarrow \mathbb{Z}_p$ and blinds $V = A_\tau^v$, computes a commitment C_j to new deposit value $ac_{j-1} - p_\tau$ and a proof that:
  - She possesses a signature on price p_τ.
  - C_j commits to $ac_{j-1} - p_\tau$.
  - C_j commits to a non-negative value [CCS08].

Transfer phase “j”: Response

$\mathcal{V}$ verifies request (POTVerReq) and computes a response (POTResp):

- $W = e(h, V)$.
- and a proof that secret key h was used to compute W.

$\mathcal{B}$ verifies response (POTVerResp) and obtains the message (POTComplete):

- $m_\tau = B_\tau / (W^{1/v}) = (\frac{m_\tau \cdot e(A_i, h)}{e(h, A_i^v)^{(1/v)}})$

Transfer phase “j”: Response

$\mathcal{V}$ verifies request (POTVerReq) and computes a response (POTResp):

- $W = e(h, V)$.
- and a proof that secret key h was used to compute W.

$\mathcal{B}$ verifies response (POTVerResp) and obtains the message (POTComplete):

- $m_\tau = B_\tau / (W^{1/v}) = \left(\frac{m_\tau \cdot e(A_i, h)}{e(h, A_i^v)^{(1/v)}} \right)$

Transfer phase “j”: Response

$\mathcal{V}$ verifies request (POTVerReq) and computes a response (POTResp):

- $W = e(h, V)$.
- and a proof that secret key h was used to compute W.

$\mathcal{B}$ verifies response (POTVerResp) and obtains the message (POTComplete):

- $m_\tau = B_\tau/(W^{1/v}) = (\frac{m_\tau \cdot e(A_i,h)}{e(h,A_i^v)^{(1/v)}})$

Outline

1. Efficient Priced Oblivious Transfer
   - Construction
   - Comparison with Previous Work
2. Optimistic Fair POT
   - Definition
   - Construction

Comparison with Previous Work

UC Secure vs Our Scheme

	[RKP09]	Our Scheme
UC	Yes	No
Standard Model	Yes	Yes
Static Corruptions	Yes	Yes
CRS	Yes	No
Assumptions	DLIN, TDH, HSDH	SDH, BDHE
Efficient	No	Yes

Efficiency

Given the upper bound of the deposit $D = d^a$.

Communication Efficiency

	[RKP09]	Our Scheme
Ciph	$(12N + 3d + 11) \cdot \lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$	$(2N + 2d + 2) \cdot \lvert\mathbb{G}\rvert + (N + 1) \cdot \lvert\mathbb{Z}_p\rvert + 2 \cdot \lvert\mathbb{G}_t\rvert$
Req	$(86 + 30a) \cdot \lvert\mathbb{G}\rvert$	$(a + 7) \cdot \lvert\mathbb{G}\rvert + (2a + 7) \cdot \lvert\mathbb{Z}_p\rvert + (a + 1) \cdot \lvert\mathbb{G}_t\rvert$
Resp	$28 \cdot \lvert\mathbb{G}\rvert$	$3 \cdot \lvert\mathbb{G}_t\rvert + \lvert\mathbb{Z}_p\rvert$

Outline

Outline

1. Efficient Priced Oblivious Transfer
   - Construction
   - Comparison with Previous Work
2. Optimistic Fair POT
   - Definition
   - Construction

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Definition

Transformation that turns any secure POT scheme into an Optimistic Fair POT scheme.

Properties:

- Third party $\mathcal{A}$ to resolve disputes.
- $\mathcal{A}$ is only involved in case of dispute (optimistic).
- $\mathcal{A}$ must be neutral to guarantee fairness.
- Privacy-properties of POT are guaranteed (even if $\mathcal{A}$ is corrupted).
  - $\mathcal{A}$ and $\mathcal{V}$ cannot learn τ.
  - $\mathcal{A}$ and $\mathcal{B}$ cannot learn non-purchased messages.
- Without harming efficiency.

Outline

1. Efficient Priced Oblivious Transfer
   - Construction
   - Comparison with Previous Work
2. Optimistic Fair POT
   - Definition
   - Construction

Verifiably Encrypted Signatures

A VES scheme consists of algorithms

- $\mathsf{Kg}(1^\kappa)$, $\mathsf{Sign}(sk, m)$ and $\mathsf{Vf}(pk, \sigma, m)$.
- $\mathsf{AdjKg}(1^\kappa)$ output a key pair (ask, apk) for $\mathcal{A}$.
- $\mathsf{Create}(sk, apk, m)$ computes a VES ω.
- $\mathsf{VesVf}(pk, apk, \omega, m)$ verifies a VES ω.
- $\mathsf{Adj}(pk, ask, apk, \omega, m)$ extracts σ form ω.

Properties:

- Unforgeability.
- Opacity.

Verifiably Encrypted Signatures

A VES scheme consists of algorithms

- Kg(1^κ), Sign(sk, m) and Vf(pk, σ, m).
- AdjKg(1^κ) output a key pair (ask, apk) for $\mathcal{A}$.
- Create(sk, apk, m) computes a VES ω.
- VesVf(pk, apk, ω, m) verifies a VES ω.
- Adj(pk, ask, apk, ω, m) extracts σ form ω.

Properties:

- Unforgeability.
- Opacity.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

Protocol based on VES

Non privacy-preserving e-commerce protocol based on VES:

- $\mathcal{B}$ requests an item and sends a VES.
- $\mathcal{V}$ sends the item.
- $\mathcal{B}$ reveals a valid signature.
- If $\mathcal{B}$ does not reveal it, $\mathcal{V}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.
- If $\mathcal{V}$ does not fulfill delivery, $\mathcal{B}$ complains.
  - $\mathcal{A}$ verifies $\mathcal{V}$ fulfills delivery.
  - $\mathcal{A}$ reveals a signature to $\mathcal{V}$.

OFPOT based on VES

- In non-privacy preserving protocols, $\mathcal{A}$ can easily verify whether $\mathcal{V}$ fulfills delivery.
- In POT $\mathcal{A}$ can learn neither $m_1, \ldots, m_N$ nor τ.
- However, correctness of requests and responses can be publicly verified.
  - POTVerReq does not need secret key sk.
  - POTVerResp does not need τ.

OFPOT based on VES

- In non-privacy preserving protocols, $\mathcal{A}$ can easily verify whether $\mathcal{V}$ fulfills delivery.
- In POT $\mathcal{A}$ can learn neither $m_1, \ldots, m_N$ nor τ.
- However, correctness of requests and responses can be publicly verified.
  - POTVerReq does not need secret key sk.
  - POTVerResp does not need τ.

OFPOT based on VES

- In non-privacy preserving protocols, $\mathcal{A}$ can easily verify whether $\mathcal{V}$ fulfills delivery.
- In POT $\mathcal{A}$ can learn neither $m_1, \ldots, m_N$ nor τ.
- However, correctness of requests and responses can be publicly verified.
  - POTVerReq does not need secret key *sk*.
  - POTVerResp does not need τ.

OFPOT based on VES

- In non-privacy preserving protocols, $\mathcal{A}$ can easily verify whether $\mathcal{V}$ fulfills delivery.
- In POT $\mathcal{A}$ can learn neither $m_1, \ldots, m_N$ nor τ.
- However, correctness of requests and responses can be publicly verified.
  - POTVerReq does not need secret key sk.
  - POTVerResp does not need τ.

OFPOT based on VES: construction

Generic OFPOT scheme

$\mathcal{V}((m_i, p_i)_{i=1}^{N})$	**Initialization**	$\mathcal{B}(ac_0)$
$sk, \forall i, C_i = \text{POTInitV}(m_i, p_i)$	$(C_1, \ldots, C_N)$ $\longrightarrow$	Verify $(C_1, \ldots, C_N)$
ac_0	$\longleftarrow$ $(ac_0, \sigma(0, ac_0))$	ac_0
$\mathcal{V}$	**Transfer "j"**	$\mathcal{B}(\tau)$
POTVerReq(Q)	$\longleftarrow$ $(Q, \omega(j, Q))$	$(Q, Qpriv) = \text{POTReq}(C_\tau)$
$R = \text{POTResp}(Q, sk)$	(R) $\longrightarrow$	POTVerResp(R)
	$\longleftarrow$ $(Q, \sigma(j, Q))$	$(m_\tau) = \text{POTComplete}(R, Qpriv)$

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

OFPOT based on VES: disputes

$\mathcal{V}$ complains:

- $\mathcal{V}$ sends request $Q, \omega(j, Q)$ and response R to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and response.
- $\mathcal{A}$ sends R to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

$\mathcal{B}$ complains:

- $\mathcal{B}$ sends request $Q, \omega(j, Q)$ to $\mathcal{A}$.
- $\mathcal{A}$ verifies request and sends it to $\mathcal{V}$.
- $\mathcal{V}$ returns a response to $\mathcal{A}$.
- $\mathcal{A}$ verifies the response.
- $\mathcal{A}$ sends response to $\mathcal{B}$ and reveals $\sigma(j, Q)$ to $\mathcal{V}$.

Conclusion

POT scheme.

- Full-simulation secure.
- Standard model.
- Efficient.

Optimistic fair POT.

- $\mathcal{A}$ only involved in case of dispute.
- Privacy preserved when $\mathcal{A}$ corrupted.
- Efficient.

Conclusion

POT scheme.

- Full-simulation secure.
- Standard model.
- Efficient.

Optimistic fair POT.

- $\mathcal{A}$ only involved in case of dispute.
- Privacy preserved when $\mathcal{A}$ corrupted.
- Efficient.

For Further Reading I

William Aiello, Yuval Ishai, and Omer Reingold.
Priced oblivious transfer: How to sell digital goods.
In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2001.

Jan Camenisch, Rafik Chaabouni, and Abhi Shelat.
Efficient protocols for set membership and range proofs.
In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

For Further Reading II

- Jan Camenisch, Gregory Neven, and Abhi Shelat.
  Simulatable adaptive oblivious transfer.
  In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590. Springer, 2007.

- Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi.
  Essentially optimal universally composable oblivious transfer.
  Cryptology ePrint Archive, Report 2008/220, 2008. http://eprint.iacr.org/.

- Shimon Even, Oded Goldreich, and Abraham Lempel.
  A randomized protocol for signing contracts.
  Commun. ACM, 28(6):637–647, 1985.

For Further Reading III

Oded Goldreich.
A simple protocol for signing contracts.
In *CRYPTO*, pages 133–136, 1983.

Steve Kremer.
Formal analysis of optimistic fair exchange protocols, 2004.

Alfredo Rial, Markulf Kohlweiss, and Bart Preneel.
Universally composable adaptive priced oblivious transfer.
In Hovav Shacham and Brent Waters, editors, *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 231–247. Springer, 2009.

For Further Reading IV

Indrakshi Ray and Indrajit Ray.
An anomymous fair exchange e-commerce protocol.
In *IPDPS*, page 172. IEEE Computer Society, 2001.

Christian Tobias.
Practical oblivious transfer protocols.
In *IH '02: Revised Papers from the 5th International Workshop on Information Hiding*, pages 415–426, London, UK, 2003. Springer-Verlag.