A Secure Protocol for Exchanging Cards in P2P Trading Card Games Based on Transferable e-cash

M. V. M. Silva and M. A. Simplicio Jr.

***Abstract*—Trading card games (TCG) distinguish from traditional card games mainly because the cards are not shared between players in a match. Instead, users play with the cards they own (e.g., purchased or traded with other players), which corresponds to a subset of all cards produced by the game provider. Even though most computer-based TCGs rely on a trusted third-party (TTP) for preventing cheating during trades, allowing them to securely do so without such entity remains a challenging task. Actually, potential solutions are related to e-cash protocols, but, unlike the latter, TCGs require users to play with the cards under their possession, not only to be able to pass those cards over. In this work, we present the security requirements of TCGs and how they relate to e-cash. We then propose a concrete, TTP-free protocol for anonymously trading cards, using as basis a secure transferable e-cash protocol.**

***Keywords*—Trading Card Games (TCG), secure trading, TTP-free, transferable e-cash.**

I. Introduction

A trading card game (TCG) is a type of card game in which, instead of using a fixed deck, each player creates his/her own deck from a subset of all cards made available by the game provider [1]. During a match, players usually do not share their cards with their opponents; hence, as any different cards may exist, part of the game is to build decks that support a target strategy or game style. To build better decks, users may either trade cards with other users or purchase them directly from the game provider. To improve their revenue, in the last years some providers have expanded their markets beyond the realm of physical cards, including digital versions of their games. This is the case, for example, of "Magic: the Gathering$^{\text{TM}}$", one of the first TCGs ever released[1].

To set matches and avoid cheating, digital TCGs typically use a client-server architecture, where the centralized system acts as card market and referee for the matches between players. When considering mobile applications, however, a peer-to-peer (P2P) architecture may present advantages over the client-server one [1], [2]. The reason is that a client-server model obliges players to have a continuous Internet connection when trading or playing, preventing them to do any of those actions otherwise. If the game protocols are designed so it does not depend on a trusted third party (TTP) to prevent cheating, on the other hand, then a local connection would be enough, bringing convenience to users.

Playing traditional card games in a P2P model was firstly proposed in *mental poker* [3] and different solutions were proposed since them (for a survey, see [4]). These works also served as basis for TTP-free solutions for TCGs, such as Match+Guardian [2] and SecureTCG [1], which allow the detection of cheating attempts during a match with two or more players. Despite those advances concerning *in-game* cheating, such protocols still depend on a trusted entity for each card trading event, leaving the task of reducing this dependence as a subject for future work.

Trading cards in a TTP-free manner is a problem that resembles that tackled by transferable e-cash protocols [5], [6], where the cards replace the digital money. For example, as in e-cash, a player should be able to anonymously trade cards with other players without the need of a TTP for mediating the transactions; however, if he/she sends the same card to two or more players (i.e., "double-spends" it), this should be detectable and the transgressor's anonymity should be revoked.Nevertheless, TCGs also have additional requirements, as there is no concept similar to "playing with owned cards" in the context of e-cash. To the best of our knowledge, there is no definition in the literature of the full set of security requirements that apply to card trading, which hinders further progress in this area.

Aiming to tackle the above issues, in this work we: (1) define the requirements for secure card trading; and (2) instantiate a protocol that fulfills those requirements, allowing players to detect cheating attempts when exchanging cards which each other even before a match starts. The propose scheme is based on existing transferable e-cash protocols (namely, [6] and [7]), with the required adaptations for allowing players to: (1) purchase cards from the game provider in a privacy-preserving manner, meaning that a card cannot be linked to any user unless its owner generates a proof of ownership; (2) use the cards they own in a match; (3) trade cards with other players; (4) verify the validity of the card without the intervention of a TTP, independently of the number of previous owners the card has ever had; (5) let the game provider know about cheating events, such as a user playing with a card that has already been handed over to another user. Since the resulting protocol is transparent to how the matches themselves are handled, it can also be integrated with in-game cheating-detection mechanisms such as the aforementioned Match+Guardian or SecureTCG, thus allowing the construction of secure P2P-based TCG environment.

The rest of this document is organized as follows. Section II discusses the characteristics of TCGs, describing its security requirements compared to those of e-cash protocols.

M. V. M. Silva, Laboratório de Arquitetura de Redes de Computadores, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, mvsilva@larc.usp.br

M. A. Simplicio Jr, Laboratório de Arquitetura de Redes de Computadores, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, mjunior@larc.usp.br

[1] http://magic.wizards.com/en/content/magic-duels

Section III presents the notation and the building blocks of the proposed protocol, as well as the corresponding security assumptions. Section IV then uses these building blocks to describe a concrete instantiation of the proposed protocol. Finally, Section VI presents our final considerations.

II. Background

This section presents the basic concepts related to TCGs, including the game architecture, the representation of the cards, and the corresponding security requirements and threats.

A. Architecture

Following the notation of [1], [2], the architecture of a P2P TCG encompasses a game server and the players.

The game server is responsible for any action that requires a trusted authority or centralized information storage. One of its primary roles is to serve as a *registration center* for players: to enroll in the system, a user must register with a unique identifier (e.g. e-mail or social security number) and provide his/her public key; the game server then generates a digital certificate to assert this information, allowing anyone to verify who are the system's authorized users.

The game server also acts as a *card market*, being responsible for selling and digitally signing cards, so the buyer can prove that a card is valid as well as its ownership. As a result, the server does not need to keep record of the cards possessed by each player, as ownership varies with time and, as proposed in this work, trading may occur without the server's knowledge. The server is also responsible for informing players of the cards available in the game, as new releases usually add several new cards to the game.

Finally, the server is also the entity that plays the role of *game auditor*, verifying claims regarding cheating attempts and eventually punishing those responsible for misbehavior. For example, in [1], [2], the players may send after-match information to the server to prove that a user cheated, e.g., by modifying the sequence or contents of their deck during a match. If a player sends to the server the list of cards employed by an adversary, the server should also be able to verify the usage of cards that were not under a malicious player's possession at the time of the match (e.g., because he/she traded it earlier). Providing such after-match data is actually very common, as this information is normally required to rank players depending on the number of victories in matches.

Any other action that does not require a TTP, such as playing the game or trading cards, can be performed in purely P2P fashion and still be protected by cheating-detection mechanisms. As in-game cheating is quite thoroughly covered in [1], in this work we focus only on cheating-detection during card trading.

B. Representation of cards

The minimal representation of a card C in a typical TCG corresponds to a tuple $C = (ID, d, V, owner)$, where: ID is the card's unique identifier; d is the card's game-specific information, which defines how it affects the game, which are the conditions for it to be played, etc.; V is some validation information, which allows any player to verify that the card was indeed issued by the game provider; and $owner$ is the information that allows the card's current owner to be identified. Since V and $owner$ are directly related to a players' ability to detect invalid cards or attempts to play with cards that are not actually owned by a player, they are described in more detail later in Section IV, in which a concrete instantiation of the proposed protocol for this purpose is described.

C. Comparison with e-cash

The security issues that appear when trading cards are somewhat similar to those faced by transferable e-cash. Indeed, both systems must provide some sort of *balance*, so that the number of elements (coins or cards) of the system should not grow without the central server's authorization. Hence, no user should be able to produce more elements than what the central server has emitted, which could be done by forging a new element or duplicating an existing one. Many actions supported by card trading and transferable e-cash protocols are also similar: stamping new cards is similar to minting new coins, while trading cards is equivalent to spending coins.

It is, thus, reasonable to build a secure card trading protocol from a transferable e-cash scheme. In this case, like coins, the card's portion that indicates ownership ($owner$), grows in size with each transference [5], or need to be stored somewhere else to prevent such growth (e.g., in a receipt [8]). To avoid indefinite growth, players may *refresh* their cards, which is equivalent to deposit a coin and get a new, mint version of it. TTP-free transferability also raises the problem of duplicating existing elements, an issue that cannot be prevented but can be detected so that the culprit is identifiable when the coin is deposited at the central server. More precisely, in case of double-spending in transferable e-cash schemes, the central server is able to revoke the anonymity of the user responsible for misbehavior, and only of that user, independently of how many owners the coin had before or after it was copied.

In the context of TCGs, however, the double spending problem is more complicated because players may not only trade, but also use their cards without transferring its ownership. Therefore, TCGs also need mechanisms for detecting a scenario in which a user irregularly plays with a card that has been previously traded. As further discussed in Section IV, this can be accomplished if the server crosses the information about refreshed cards with those received from match reports. Hence, refreshing cards benefits both honest players and the game server: the former get a shorter copy of the card, which is less computationally expensive to verify and trade, while the latter is able to audit trades by using the information stored in the cards submitted for refreshing. The same mutual benefit applies to the match reports: honest players who win matches can raise their ranks by informing their victories to the server; honest players who lose matches can make sure the opponent played fairly; and the server can audit if some refreshed or traded card has been illicitly used in a match. It should, thus, be quite easy to encourage players to provide such information often to the server.

In summary, five types of cheating can appear when cards are traded: (1) *Double-refresh*: refreshing a same card twice, obtaining several valid instances of the same card but purchasing a single one; (2) *Double-trade*: sending copies of the same card to different users; (3) *Trade-then-play*: playing with a card that has already been passed to another user; (4) *Refresh-then-trade*: refreshing a card C to obtain a mint version of it, C', but then trading copies of C with other users; and (5) *Refresh-then-play*: refreshing a card C to obtain a mint version of it, C', but then using C in matches with other players.

D. System requirements

From the previous discussion, we can postulate that the following security and usability requirements must be met in by secure P2P-based TCG system.

Verifiable stamping: The card market must stamp cards, so their validity and ownership can be verified without the need of contacting the central server.

TTP-free transferability: Players should be able to trade cards with each other without the intervention of a TTP, and the new ownership can also be verified without the need of contacting a trusted server.

Anonymity: Suppose that U_0 purchases a given card C, and then that card is repeatedly traded among a set of users $\{U_{1...n}\}$ before the last owner, U_{n+1}, informs the server about this ownership. In this case, the server only learns the identity of U_{n+1}, while the C's previous owners remain anonymous. In addition, during this process user U_i only learns the identity of U_{i-1} and U_{i+1}.

Balance: The number of cards in the system cannot grow unless the central server stamps new cards, with invalid duplicates being detected and removed.

Cheat detection: Players cannot trade a card more than once without losing their anonymity toward the server, nor play with a card after having traded it.

Exculpability: The game server, even if in collusion with users, cannot falsely prove that an honest user has cheated, i.e., the cheating-detection mechanism only allows identifying users who have duplicated a card (either for trading or playing with it).

III. Building blocks

This section presents the mechanisms necessary for a concrete construction of a P2P TCG trading protocol. Specifically, the proposed scheme is based on the transferable e-cash scheme described in [6] and revisited in [7], which relies on asymmetric pairings, witness-indistinguishable non-interactive proofs, verifiable random functions and structure-preserving blind signatures.

A. Preliminaries and Notation

Assume three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order q, and a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ having the following properties: (1) *bilinearity*: $\forall G \in \mathbb{G}_1, H \in \mathbb{G}_2, a, b \in \mathbb{Z}_q : e(G^a, H^b) = e(G, H)^{ab}$; (2) *non-degenerative*: $\forall G \neq 1_{\mathbb{G}_1}, H \neq 1_{\mathbb{G}_2} : e(G, H) \neq 1_{\mathbb{G}_T}$; and (3) e is *efficiently computable*. The pairing parameters $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G, H, e)$ are a Type-3 (or asymmetric) pairing if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

In a finite set $\mathcal{S}$, $s \xleftarrow{\$} \mathcal{S}$ denotes that s is sampled uniformly at random from $\mathcal{S}$. If some protocol $\mathcal{R}$ is a multi-party algorithm between parties $\mathcal{A}$ and $\mathcal{B}$, then $\mathcal{R}\big(\mathcal{A}(a) \leftrightarrow \mathcal{B}(b)\big)$ is the execution of $\mathcal{R}$ with inputs a from $\mathcal{A}$ and b from $\mathcal{B}$. We also consider a cryptographic hash function $\mathcal{H} : \{0, 1\}^* \to \{0, 1\}^q$ (e.g., SHA-3 [9]). If $\mathcal{H}$ has more than one input, we consider the inputs are concatenated in the order they are presented. We also define a set of map functions from $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{Z}_q$ to $\{0, 1\}^*$, so that group elements can be used as input to the hash function.

A *co-* security assumption is the translation of an assumption from a symmetric pairing to an asymmetric pairing. The superscript numbers in these assumptions are the necessary number of duplicated elements from any source group of the pairing for validation of security, as discussed in details in [10].

B. Groth-Sahai proofs

Proofs of knowledge allow a party to prove knowledge of some secret value without revealing it, which is done by showing a witness satisfying some relation that depends on the secret. The most efficient proofs are usually interactive, based on a challenge-response method. For transferable elements, however, one cannot expect any interaction with the parties not directly involved in the current transference. Nevertheless, as shown in [11], non-interactive proofs can still be performed in an efficient manner when the relation to be proved is a set of equations in some defined format and the witnesses are variables that belong to the solutions set. Such Groth-Sahai proofs depend on a signature scheme, and the relation is defined by the verification equation. For a structure-preserving signature, which is adopted in this article, the relation is the following pairing product equation (PPE):

$$\prod_{i=0}^{m} e(X_i, B_i) \prod_{j=0}^{n} e(A_j, Y_j) \prod_{i=0}^{m} \prod_{j=0}^{n} e(X_i, Y_j)^{\gamma_{ij}} = t \tag{1}$$

where $A_j \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $t \in \mathbb{G}_T$ and $\gamma_{ij} \in \mathbb{Z}_q$ are constants, and $X_i \in \mathbb{G}_1$ and $Y_j \in \mathbb{G}_2$ are variables. For a proof, we need 4 elements in $\mathbb{G}_1$ and 4 in $\mathbb{G}_2$ for each equation, whereas each variable will be committed to 2 elements in their group.

The algorithms employed by a Groth-Sahai proof are: $GSCommit(x, open) \to C$, that commits the variables to be used in a proof; $GSProve(\{x_i \ in \ C_i\}_{i=1..n} | eq) \to \phi$, that creates a proof that the prover knows witnesses that satisfies the equation; and $GSVerify(\phi, eq) \to \{0, 1\}$, that verifies if the proof is valid.

We refer to [11] for a concrete instantiation under Symmetric External Diffie-Hellman (SXDH) assumption [12].

C. Verifiable random function

A verifiable random function (VRF) f_s is a especial type of pseudorandom function that allows anyone who knows the secret seed s to compute $f_s(x)$ for any x and also to prove that $f_s(x)$ is indeed correct without compromising the unpredictability of f_s at any point $x^{'} \neq x$ [13]. Of especial interest to this work is the VRF instantiation described in [7], in which the verification of x uses the PPE $e(Y = G^{\frac{1}{s+x}}, H^s \cdot H^x) = e(G, H)$, so knowledge of s and x can be proved by the Groth-Sahai method. This specific instantiation is secure under the q-Decisional Diffie-Hellman inversion (q-DDHI) assumption (in $\mathbb{G}_1$) [14] and SXDH (for the Groth-Sahai proof).

D. Structure-preserving blind signature

Blind signatures were originally proposed in the context of anonymous e-cash [15], allowing a user to obtain a valid signature on values unknown to the signer. If transferability is required, the user doing the transfer also needs to prove knowledge of the signed values, which can be achieved using *structure-preserving* (or automorphic) signatures [16]. In such signature schemes, the verification keys lie in the message space, the messages and signatures comprise elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, and the verification is done using a set of PPEs. Using the set of signatures, a prover can create a non-interactive proof of knowledge that some witnesses satisfy the PPE for verification.

There are few structure-preserving blind signature schemes in the literature, and even fewer for efficiently signing a set of messages. For the purposes of this work, we adapt the P-signature scheme proposed by [17], converting it to an asymmetric pairing setting by means of the method proposed in [10]. The reason for this modification is that, even though [17] is quite efficient, it uses symmetric pairing and supersingular elliptic curves, requiring fields of larger size to achieve a security level similar to what can be obtained with an asymmetric pairing [18]. The resulting scheme comprises the following operations:

- $PSetup(k) \rightarrow pparams$: Generates the set of public parameters $pparams$ for the signature, which corresponds to the parameters of Groth-Sahai proofs under an asymmetric pairing setting. For the sake of simplicity, these parameters are omitted in the descriptions of the remainder operations.
- $PKeyGen(n) \rightarrow (pk, sk)$: Choose $\alpha, \beta, \gamma, \omega \xleftarrow{\$} \mathbb{Z}_q^*$ and $U, U_0 \xleftarrow{\$} \mathbb{G}_2$. Compute $U_1 = G^\beta, \Omega = G^\omega, A = H^\gamma$, as well as $\forall i \in [1, 2n] \setminus (n+1) : G_i = G^{\alpha^i}, H_i = H^{\alpha^i}$, to sign n messages. Output the private key $sk = (\gamma, \omega, \beta)$ and the public key $pk = (U, U_0, U_1, \Omega, A, \{G_i\}_{i=1..n}, \{H_i\}_{i=1..n})$.
- $PSign(sk, \vec{m}) \rightarrow \sigma$: For $\vec{m} = (m_1, ..., m_n)$, pick $r \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $K = H^r \cdot \prod_{j=1}^n H_{n+1-j}^{m_j}$. Choose $c \xleftarrow{\$} \mathbb{Z}_q^*$ and compute: $\sigma_1 = H^{\gamma/(\omega+c)}$, $\sigma_2 = G^c$, $\sigma_3 = U^c$, $\sigma_4 = (U_0 \cdot K^\beta)^c$, $\sigma_5 = K^c$, $\sigma_6 = K$, $\sigma_r = r$.
Output the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_r)$.
- $PVerifySig(pk, \sigma, \vec{m}) \rightarrow \{0, 1\}$: Return 1 if and only if the following equations hold: $e(G, A) = e(\Omega \cdot \sigma_2, \sigma_1)$, $e(\sigma_2, U) = e(G, \sigma_3)$, $e(G, \sigma_4) = e(\sigma_2, U_0) \cdot e(U_1, \sigma_5)$, $e(G, \sigma_5) = e(\sigma_2, \sigma_6)$, and $\sigma_6 = H^r \cdot \prod_{j=1}^n H_{n+1-j}^{m_j}$.
- $PCommit(pk, \vec{m}) \rightarrow (K, r)$: Choose $r \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $K = H^r \cdot \prod_{j=1}^n H_{n+1-j}^{m_j}$. Output the commitment $comm = (K, r)$.
- $PUpdateComm(pk, \vec{m}, K) \rightarrow K'$: Compute and output $K' = K \cdot \prod_{j=0}^n H_{n+1-j}^{m_j}$.
- $PWitGen(pk, i, \vec{m}, K, r) \rightarrow W_i$: If K is a commitment to $\vec{m}$ with opening r, compute and output $W_i = G_i^r \cdot \prod_{j=1;j\neq i}^n G_{n+1+i-j}^{m_j}$.
- $PVerifyWit(pk, i, m_i, W_i, K) \rightarrow \{0, 1\}$: Return 1 if and only if the following equation holds: $e(G_i, K) = e(G_1, H_n)^{m_i} \cdot e(W_i, H)$.
- $PProveCom(pk, \vec{m}, K, r) \rightarrow \phi_K$: Generate witnesses for each message committed, $\forall i \in [1, n] : W_i = WitGen(pk, i, \vec{m}, K, r)$. Generate a Groth-Sahai proof of knowledge that the following pairing product equations are valid (group elements in bold are constants): $\forall i \in [1, n] : e(\mathbf{G_i^{-1}}, K) \cdot e(\mathbf{G_n}, H_1^{m_i}) \cdot e(W_i, \mathbf{H}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$; $\forall i \in [1, n] : e(\mathbf{G_1}, H^{m_i}) \cdot e(\mathbf{G^{-1}}, H_1^{m_i}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$; $\forall i \in [1, n] : e(\mathbf{G_{2n}}, H^{m_i}) \cdot e(\mathbf{G^{-1}}, H_{2n}^{m_i}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$. Output the proof ϕ_K and the complementary commitments (from the Groth-Sahai system), $\forall i \in [1, n] : C_{m_i H_1}, C_{m_i H}, C_{m_i H_{2n}}, C_{W_i}; C_K$.
- $PVerifyProofCom(\phi_K) \rightarrow \{0, 1\}$: Verify if the Groth-Sahai proof of knowledge ϕ_K was correctly constructed.
- $\Big(PObtainSig(pk, \vec{m}_P) \leftrightarrow PIssueSig(sk, \vec{m}_S)\Big) \rightarrow \sigma$:
  - The User commits the message $\vec{m}_P$ as $(K, r') = PCommit(pk, \vec{m}_P)$, then sends K to the Signer with a proof of knowledge $\phi_K = PProveCom(pk, \vec{m}_P, K, r')$ that the commitment is valid.
  - The Signer verifies the proof of knowledge with a call to $PVerifyProofCom(\phi_K)$, updates the commitment to $K' = PUpdateCom(pk, \vec{m}_S, K)$, and blindly signs the commitment with random seeds $c, r'' \xleftarrow{\$} \mathbb{Z}_q^*$ as: $\sigma_1 = H^{\gamma/(\omega+c)}$, $\sigma_2 = G^c$, $\sigma_3 = U^c$, $\sigma_4 = (U_0 \cdot (K' \cdot H^{r''})^\beta)^c$, $\sigma_5 = (K' \cdot H^{r''})^c$, $\sigma_6 = K' \cdot H^{r''}$, and $\sigma'_r = r''$. The Signer then sends $\sigma' = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma'_r)$ to the User.
  - The User updates $\sigma_r = r' + \sigma'_r$ and outputs $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_r)$.
- $PProveSig(pk, \vec{m}, \sigma) \rightarrow \phi_\sigma$: Parse $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_r)$. Generate witnesses for each message signed $\forall i \in [1, n] : W_i = WitGen(pk, i, \vec{m}, \sigma_6, \sigma_r)$. Generate a Groth-Sahai proof of knowledge that the following pairing product equations are valid (the group elements in bold are constants):
  - Signature equation validation: $e(\mathbf{\Omega}, \sigma_1) \cdot e(\sigma_2, \sigma_1) \cdot e(\mathbf{G}, A^{-1}) = \mathbf{1}_{\mathbb{G}_\mathbb{T}}$; $e(\sigma_2, \mathbf{U_0}) \cdot e(\mathbf{U_1}, \sigma_5) \cdot e(\mathbf{G^{-1}}, \sigma_4) = \mathbf{1}_{\mathbb{G}_\mathbb{T}}$; $e(\sigma_2, \mathbf{U}) \cdot e(\mathbf{G^{-1}}, \sigma_3) = \mathbf{1}_{\mathbb{G}_\mathbb{T}}$; $e(\mathbf{G}, \sigma_5) \cdot e(\sigma_2, \sigma_6)^{-1} = \mathbf{1}_{\mathbb{G}_\mathbb{T}}$.
  - Message pertinence validation: $\forall i \in [1, n] : e(\mathbf{G_i^{-1}}, \sigma_6) \cdot e(\mathbf{G_n}, H_1^{m_i}) \cdot e(W_i, \mathbf{H}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$; $\forall i \in [1, n] : e(\mathbf{G_1}, H^{m_i}) \cdot e(\mathbf{G^{-1}}, H_1^{m_i}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$; $\forall i \in [1, n] : e(\mathbf{G_{2n}}, H^{m_i}) \cdot e(\mathbf{G^{-1}}, H_{2n}^{m_i}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$.
  - Equality commitment validation: $e(G, \mathbf{A}) \cdot e(G, A^{-1}) = \mathbf{1}_{\mathbb{G}_\mathbf{T}}$; $e(G, \mathbf{H}) = \mathbf{e(G, H)}$

Output the proof ϕ_σ and the complementary commitments (from the Groth-Sahai system) $\forall i \in [1, n] : C_{m_i H_1}, C_{m_i H}, C_{m_i H_{2n}}, C_{W_i}; \forall j \in [1, 6] : C_{\sigma_j}; C_{-A}, C_G$.

- $PVerifyProofSig(\phi_\sigma) \rightarrow \{0, 1\}$: Verify if the Groth-Sahai proof of knowledge ϕ_σ was correctly constructed.

Table I lists the number of elements necessary for the signature scheme when signing n messages.

TABLE I
NUMBER OF ELEMENTS FROM EACH GROUP WHEN SIGNING n MESSAGES

Object	$\mathbb{Z}_q$	$\mathbb{G}_1$	$\mathbb{G}_2$	*Object*	$\mathbb{Z}_q$	$\mathbb{G}_1$	$\mathbb{G}_2$
Private key (sk)	3	0	0	Opening ($open$)	1	0	0
Public key (pk)	0	$2+n$	$3+n$	Signature (σ)	1	1	5
Message ($\vec{m}$)	n	0	0	Proof of commitment (ϕ_K)	0	$4+2n$	$6+6n$
Commitment (K)	0	0	1	Proof of signature (ϕ_σ)	0	$8+2n$	$18+6n$

E. Compact e-cash

The compact e-cash scheme originally described in [6] and revised in [7] allows a user to withdraw several coins (i.e., a wallet) within a single message. In the context of TCGs, this scheme is interesting because it (1) allows several seed parameters (instead of coins) to be signed altogether and (2) it provides a direct method for identifying cheaters, who have their public key recovered, so the server do not need to screen the whole user database in search for the culprit. The version actually adopted in the proposed solution is based on the adaptation from [19], which achieves transferability with strong anonymity, by means of the following operations (for a concrete instantiation and details, see [7]):

- $Setup$: The bank generates a public/private key pair and publishes its public key together with the system's public parameters.
- $Register$: The user randomly generates a public/private key pair based on the system parameters and retrieves a certificate from the bank for the public key generated in this manner. The bank stores the user's identity and corresponding public keys, which allows users to be identified in case of double-spending.
- $Withdraw$: The user produces seed values and commits them to the bank, which in turn blindly signs those values. This creates a new anonymous wallet with as many coins as the number of seeds provided.
- $Spend$: Users may exchange either unspent coins from their wallets or coins previously received. In the former case, the user creates a new coin from the serial seed and treats it just like a received coin. Each time a coin is spent, a tag giving ownership of it to the receiver is added to the coin representation, making it grow in size. All tags must be verified by the receiver to ensure the previous transaction are valid and, thus, that the coin actually hold value.
- $Deposit$: The user sends the coin to the bank, which verifies if this coin had already been deposited. If it has, the bank verifies if this is a case of double-deposit (i.e., if the user is trying to deposit the same coin twice) or of double-spending (i.e., if it was sent to two different users at some point in time).
- $Identify$: In case of double-spending, the bank retrieves the public key of the perpetrator, so the required administrative penalties can be applied.

A wallet $W = (sk_U, s, t, \sigma)$ is composed by the private key sk_U of the owner, a serial seed s, a transfer seed t, and a signature σ on these values. A coin $C = (S, \phi_S, \phi_\sigma, \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\})$ is identified by a serial number S and its proof of validity ϕ_S, proof of knowledge on the signature of the wallet ϕ_σ, and a set π_T of j transferences. Each transference is composed by a transference tag number T_j and its proof of validity ϕ_{T_j}, a tag of ownership r_j, and some public information i_j. When a coin is spent, a new tag indicating the transference of ownership is inserted into π_T.

The serial number S is picked at random to provide a unique identifier for each coin. It is then employed in the *serial number generation* function f_S, a VRF that is defined by Equation 2. In this equation, s is a seed signed in the wallet and sk_U is the private key of the owner (or the index of the coin, if more than one coin can be withdrawn).

$$f_S(sk_U, s) = G^{\frac{1}{s+sk_U}} \tag{2}$$

The transference tag T identifies each transference, also picked at random. It is then employed in a modified version of the VRF, the *transference tag generation* function f_T described in Equation 3. In this equation, t is a seed signed in the wallet or referenced by previous transference, sk_U is the secret key of the owner and R is the hash of the private (that contains the owner) and public (e.g., a timestamp) information of the transference.

$$f_T(sk_U, t, R) = \left(G^R\right)^{sk_U} G^{\frac{1}{t+sk_U}} \tag{3}$$

Finally, the ownership tag r is a randomly-picked value used to hide the private key of the coin's owner. Similarly to S, it is employed in a VRF, the *ownership tag generation* function f_r from Equation 4, where sk_U is the private key of the owner and i is some public information related to the transference. This tag is used to create the transference tag that allows the owner of the coin to prove that the last transference was directed to him/her, so this information is used to compute R, linking the transference tag T to the owner, represented by r.

$$f_r(sk_U, i) = G^{\frac{1}{sk_U+i}} \tag{4}$$

The revised version also presents a proof of knowledge protocol for the serial number generation $\Phi_S : \big(Prove(sk_U, s) \rightarrow \phi_S; Verify(\phi_S) \rightarrow \{0,1\}\big)$, for the transference tag $\Phi_T : \big(Prove(sk_U, t, R) \rightarrow \phi_T; Verify(\phi_T) \rightarrow \{0,1\}\big)$ and for the ownership tag $\Phi_r : \big(Prove(r, i) \rightarrow \phi_r; Verify(\phi_r) \rightarrow \{0,1\}\big)$, based on Groth-Sahai proofs which proves the values presented in the coin (S, T and R) were computed using the respective functions. Due to space limitation, we refer the reader to [7] for details.

IV. PROPOSED PROTOCOL

In this section we present a concrete instantiation of the proposed scheme for secure trading cards, using the building blocks described in Sec. III. The roles of registration center $\mathcal{C}$, card market $\mathcal{M}$ and game auditor $\mathcal{A}$ are played by the game server $\mathcal{G} = \mathcal{C} \cup \mathcal{M} \cup \mathcal{A}$. A card C is represented by the tuple $C = (ID, d, V, owner)$, where: $ID \in \mathbb{G}_1$ is its unique identifier; $d \in \mathbb{Z}_q$ is the numeric representation of the card's description using some suitable encoding; $V = (\phi_{ID}, \phi_\sigma)$, where ϕ_{ID} and ϕ_σ are, respectively, proofs of knowledge of the construction of the ID and of the signature from the market; and $owner = \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\}$ corresponds to the records of all owners of the cards, so that, for each index j in π_T, T_j is the transference tag with proof of knowledge of the construction ϕ_{T_j}, r_j is the ownership tag and i_j is the public information regarding the transference.

The operations comprised by the proposed scheme are, then:

- $Setup()$: The game server generates the system parameters $tcgparams = (pparams_C, pparams_M)$ where $pparams_C$ and $pparams_M$ are the parameters of two signature schemes, the first to register new players and the second to stamp new cards. Both of them contain parameters of a Groth-Sahai proof system, defined over an asymmetric pairing Λ. These parameters are used by the subsequent operations and, for shortness, are omitted in their descriptions. The game server also generates two key-pairs: $(sk_C, pk_C) \leftarrow PKeyGen()$ to register players and $(sk_M, pk_M) \leftarrow PKeyGen()$ to stamp cards. It then publishes $tcgparams$, pk_C and pk_M.
- $Register(id_P[, sk_P])$: Player $\mathcal{P}$ with identity id_P generates a secret key $sk_P \xleftarrow{\$} \mathbb{Z}_q$ and computes the public key $pk_P = e(G, H)^{sk_P}$. $\mathcal{P}$ generates a proof of knowledge $\phi_P = GSProof\big(H^{sk_P}\ in\ C_{sk_P}, \theta = 1 | e(G^\theta, H^{sk_P}) = pk_P \wedge e(G^\theta, H) = e(G, H)\big)$. The triple (id_P, pk_P, ϕ_P) is sent to the registration center $\mathcal{C}$. If the proof ϕ_P is valid, $\mathcal{C}$ generates a signature $\sigma_P = PSign(sk_C, \{id_P, pk_P\})$. $\mathcal{P}$ can then present σ_P as his/her certificate.
- $Stamp\Big(\mathcal{P}(sk_P, pk_M, d) \leftrightarrow \mathcal{M}(sk_M, pk_M, d)\Big)$: To purchase an instance of a card with description d, player $\mathcal{P}$ generates a partial identifier seed $s' \xleftarrow{\$} \mathbb{Z}_q$ and a transference seed $t \xleftarrow{\$} \mathbb{Z}_q$, and the card market $\mathcal{M}$ generates the card's partial identifier component $s'' \xleftarrow{\$} \mathbb{Z}_q$. Both parties execute the interactive protocol to obtain a blind signature $\sigma_s = \Big(PObtainSig(pk_M, \{sk_P, s', t, 0\}) \leftrightarrow PIssueSig(sk_M, \{0, s'', 0, d\})\Big)$ that is returned to $\mathcal{P}$ together with s''. The player then generates a proof of knowledge $\phi_\sigma = PProveSig(pk_M, \{sk_P, s = s' + s'', t, d\}, \sigma_s)$. After that, $\mathcal{P}$ chooses some unique the public information $i_0 \leftarrow \{0,1\}^*$ (e.g., a timestamp) and computes $r_0 = G^{\frac{1}{sk_P + \mathcal{H}(i_0)}}$ and $R_0 = \mathcal{H}(r_0, i_0)$. $\mathcal{P}$ then generates the unique identifier $ID = f_S(sk_P, s)$ and the transference tag $T_0 = f_T(sk_P, t, R_0)$, together with proofs of knowledge $\phi_{ID} = \Phi_S.Prove(sk_P, s)$ and $\phi_{T_0} = \Phi_T.Prove(sk_P, t, R_0)$ of the construction, associated with the commitments in the proof of signature ϕ_σ. Finally, the player stores the card $C = (ID, d, \phi_{ID}, \phi_\sigma, \pi_T = \{T_0, \phi_{T_0}, r_0, i_0\})$.
- $Send\Big(\mathcal{P}_1(sk_{P_1}, pk_{P_2}, C) \leftrightarrow \mathcal{P}_2(sk_{P_2}, pk_{P_1})\Big)$: The receiver $\mathcal{P}_2$ chooses some public information $i \leftarrow \{0,1\}^*$ and computes $r = G^{\frac{1}{sk_{P_2} + \mathcal{H}(i)}}$ and a proof of validity $\phi_r = \Phi_r.Prove(sk_{P_2}, r, \mathcal{H}(i))$. $\mathcal{P}_2$ then sends the tuple (i, r, ϕ_r) to the current card hold, $\mathcal{P}_1$. $\mathcal{P}_1$ parses $C = (ID, d, \phi_{ID}, \phi_\sigma, \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\}_{j=0\ldots h})$ and verifies the proof of validity $\Phi_r.Verify(\phi_r, pk_{P_2})$. If everything is correct, $\mathcal{P}_1$ first sets $i_{h+1} = i$ and $r_{h+1} = r$, and then computes $R_{h+1} = \mathcal{H}(r_{h+1}, i_{h+1})$ and $t = \mathcal{H}(S, \{T_j\}_{j=0\ldots h})$. Finally, $\mathcal{P}_1$ generates a new transference tag $T_{h+1} = f_T(sk_{P_1}, t, R_{h+1})$, as well as a proof of knowledge $\phi_{T_{h+1}} = \Phi_T.Prove(sk_{P_1}, t, R_{h+1})$ of the construction. The card $C' = (ID, d, \phi_{ID}, \phi_\sigma, \pi'_T = \{T_j, \phi_{T_j}, r_j, \phi_{r_j}, i_j\}_{j=0\ldots(h+1)})$ is sent to $\mathcal{P}_2$. Upon reception, $\mathcal{P}_2$ verifies the construction of the unique identifier ID by $\Phi_S.Verify(\phi_{ID})$ and the tags $\{T_j\}_{j=0..h}$ by $\bigwedge_{j=0}^{h+1} \Phi_T.Verify(\phi_{T_j})$, as well as that the proof of ownership $\phi_{r_{h+1}}$ is valid in respect to the public key pk_{P_2} by $\Phi_r.Verify(\phi_{r_{h+1}}, pk_{P_2})$. If all proofs are correct, $\mathcal{P}_2$ stores the card C' as his/her own.
- $Play\Big(\mathcal{P}_1(sk_{P_1}, C) \leftrightarrow \mathcal{P}_2(pk_{P_1})\Big)$: Player $\mathcal{P}_1$ prepares a card $C = (ID, d, \phi_{ID}, \phi_\sigma, \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\}_{j=0..h})$ that has been updated h times. $\mathcal{P}_1$ chooses some arbitrary public information $i_{h+1} \leftarrow \{0,1\}^*$ and computes $r_{h+1} = G^{\frac{1}{sk_{P_1} + \mathcal{H}(i_{h+1})}}$, $R_{h+1} = \mathcal{H}(r_{h+1}, i_{h+1})$ and $t = \mathcal{H}(S, \{T_j\}_{j=0\ldots h})$. Then $\mathcal{P}_1$ generates a new transference tag $T_{h+1} = f_T(sk_{P_1}, t, R_{h+1})$, together with proof of knowledge $\phi_{T_{h+1}} = \Phi_T.Prove(sk_{P_1}, t, R_{h+1})$ of the construction. The card C is updated to $C' = (ID, d, \phi_{ID}, \phi_\sigma, \pi'_T = \{T_j, \phi_{T_j}, r_j, \phi_{r_j}, i_j\}_{j=0\ldots(h+1)})$ $\mathcal{P}_1$ also prepares two proofs of knowledge $\phi_{r_h} = \Phi_r.Prove(sk_{P_2}, \mathcal{H}(i_h))$ and $\phi_{r_{h+1}} = \Phi_r.Prove(sk_{P_2}, \mathcal{H}(i_{h+1}))$ to prove that the card was correctly prepared The triple $(C', \phi_{r_h}, \phi_{r_{h+1}})$ is sent to the match's opponent $\mathcal{P}_2$. Upon reception of C', $\mathcal{P}_2$ verifies the construction of the unique identifier ID by $\Phi_S.Verify(\phi_{ID})$ and transference tags $\{T_j\}_{j=0\ldots(h+1)}$ by $\bigwedge_{j=0}^{h+1} \Phi_T.Verify(\phi_{T_j})$, and that both proofs of ownership ϕ_{r_h} and $\phi_{r_{h+1}}$ are valid in respect to the public key pk_{P_1} by $\Phi_r.Verify(\phi_{r_h}, pk_{P_1}) \wedge \Phi_r.Verify(\phi_{r_{h+1}}, pk_{P_1})$. If they are all valid, $\mathcal{P}_2$ then stores this card locally, so it can report this information to the game server later, and uses the unique identifier ID to identify this card during the match.
- $Report\Big(\mathcal{P}(C) \leftrightarrow \mathcal{A}(\mathcal{RS})\Big)$: Player $\mathcal{P}$ sends to the game auditor $\mathcal{A}$ a card $C = (ID, d, \phi_{ID}, \phi_\sigma, \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\}_{j=0\ldots h})$ that an opponent has used in some match. $\mathcal{A}$ stores C in the set of reported cards $\mathcal{RS}$ and verifies if there is any card $\bar{C}$ with identifier $\bar{ID} = ID$ already reported in $\mathcal{RS}$. For each card $\bar{C}$, $\mathcal{A}$ executes $Identify(C, \bar{C})$, retrieving the list of public keys of users who had illegally duplicated this card.
- $Refresh\Big(\mathcal{P}(sk_P, C) \leftrightarrow \big(\mathcal{G} = \mathcal{A}(\mathcal{RS}) \cup \mathcal{M}(sk_M, pk_P)\big)\Big)$: Player $\mathcal{P}$ prepares a card $C = (ID, d, \phi_{ID}, \phi_\sigma, \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\}_{j=0\ldots h})$ that has been updated h times. $\mathcal{P}$ chooses some public information $i_{h+1} \leftarrow \{0,1\}^*$ (e.g., a timestamp) and computes $r_{h+1} = G^{\frac{1}{sk_P + \mathcal{H}(i_{h+1})}}$, $R_{h+1} = \mathcal{H}(r_{h+1}, i_{h+1})$ and $t = \mathcal{H}(S, \{T_j\}_{j=0\ldots h})$. Then $\mathcal{P}$ generates a new transference tag $T_{h+1} = f_T(sk_P, t, R_{h+1})$, together with proof of knowledge $\phi_{T_{h+1}} = \Phi_T.Prove(sk_P, t, R_{h+1})$ of the construction. The card C is updated to $C' = (ID, d, \phi_{ID}, \phi_\sigma, \pi'_T = \{T_j, \phi_{T_j}, r_j, i_j\}_{j=0\ldots(h+1)})$ and is sent to the game server $\mathcal{G}$. $\mathcal{A}$ stores C' in $\mathcal{RS}$ and verifies if there is any card $\bar{C}$ with identifier $\bar{ID} = ID$ already reported in

$\mathcal{RS}$. For each card $\bar{C}$, $\mathcal{A}$ executes $Identify(C', \bar{C})$, retrieving the list of public keys of users who had illegally duplicated this card. If identifying C' did not return any transgressor, both parties execute $Stamp\big(\mathcal{P}(sk_P, pk_M, d) \leftrightarrow \mathcal{M}(sk_M, pk_P)\big)$ to produce a fresh card C'' to $\mathcal{P}$.

- $Identify(C, \bar{C})$: The game auditor $\mathcal{A}$ parses cards $C = (ID, d, \phi_{ID}, \phi_\sigma, \pi_T = \{T_j, \phi_{T_j}, r_j, i_j\}_{j=0 \dots h})$ and $\bar{C} = (\bar{S}, \bar{\phi_{ID}}, \bar{\phi_\sigma}, \bar{d}, \bar{\pi_T} = \{\bar{T_j}, \bar{\phi_{T_j}}, \bar{r_j}, \bar{i_j}\}_{j=0 \dots \bar{h}})$ with the same identifier $ID = \bar{ID}$. It searches for the first index l in which $T_l \neq \bar{T_l}$, computes $R_l = \mathcal{H}(r_l, i_l)$ and $\bar{R_l} = \mathcal{H}(\bar{r_l}, \bar{i_l})$, and retrieves the public key of the perpetrator $\mathcal{D}$ as $pk_D = \big(\frac{T_l}{\bar{T_l}}\big)^{\frac{1}{R_l - \bar{R_l}}}$. If the index l is larger than the number of hops for any card (h or $\bar{h}$), this card had already been reported but had not been duplicated, so the output is empty.

The requirements of a secure card trading system, as presented in Sec. II, are fulfilled by the underlying e-cash scheme. Namely, the signature on stamping method guarantees verifiability ("own" property), anonymity when stamping (the signer cannot link signatures to new cards), and balance (if the signature is unforgeable, a new card cannot be inconspicuously created without authorization by the card market). The proof of knowledge provides transferability on trading and ad-hoc playing, given its non-interactivity property. It also keeps anonymity when trading, since it is witness-indistinguishable together with the VRF. Finally, the identification method of the e-cash scheme guarantees balance, cheat detection and exculpability.

V. Preliminary efficiency analysis

Signing, $n = 4$ messages ($\{sk_P, s, t, d\}$) with the presented P-signature scheme, a signature proof (ϕ_σ) requires 20 elements in $\mathbb{G}_1$ and 42 in $\mathbb{G}_2$ (see Table I). For the serial number generation proof (ϕ_{ID}), we need 24 elements in $\mathbb{G}_1$ and 26 in $\mathbb{G}_2$. For the transference tag generation proof (ϕ_T), we need 36 elements in $\mathbb{G}_1$ and 38 in $\mathbb{G}_2$. A card C is composed by: the unique identifier ID (1 element in $\mathbb{G}_1$, output from f_S) the proofs of knowledge of ID (ϕ_{ID}) and of the signature from the market (ϕ_σ), and a set of transferences (π_T), updated with each trade or play. Each transference j in the set is in turn composed by: the transference tag T_j (1 element in $\mathbb{G}_1$, output from f_T), the corresponding proof of knowledge ϕ_{T_j}, the ownership tag r_j (1 element in $\mathbb{G}_1$, output from f_r), and additional public information i_j, which may have variable length. Hence, for a total of t transferences and/or usages, a card needs $45 + 38t$ elements in $\mathbb{G}_1$ and $68 + 38t$ in $\mathbb{G}_2$.

The execution time is likely dominated by the pairing computations. Using a Groth-Sahai proof of knowledge, each time a card is traded or used in a match, the total execution cost corresponding basically to the 148 underlying pairing computations. As each pairing is expected to take on the order of 1 ms to run with [20] (Intel Core i5 1,6 GHz, 128-bit security level), the total time would be around 150 ms per card traded or played. We note that, while these timings are quite reasonable for trading, they may be somewhat cumbersome when playing with a deck having roughly 50 cards, as it is common in commercial TCGs, since the verification of a deck would take around 7.5 min. Nevertheless, it is important to have in mind that the preparation of a deck can be done beforehand, much before the match starts; in addition, the verification of the corresponding proofs of knowledge can happen in background during the match, which usually takes several minutes. Therefore, in practice those costs can be made transparent to players.

VI. Conclusions

In this paper, we presented the set of requirements for allowing secure trades in P2P TCGs, defining the cheating types that need to be detected. We then adapted a transferable e-cash protocol for creating a concrete scheme that fulfills those requirements The proposed scheme is based on the P-signatures described in [17], which allows a vector of messages to be signed, which is combined with a compact blind signature scheme in the asymmetric pairing setting to allow a more memory-efficient representation.

According to our preliminary analysis, the scheme is quite efficient to be used in practice, especially considering that the most expensive operations involved (namely, validating an entire deck of cards) can be performed in background, either before or during a match.

Acknowledgments

This work was supported by the São Paulo Research Foundation (FAPESP) under grant 2011/21592-8 and by the National Counsel of Technology and Scientific Development (CNPq) under grants 482342/2011-0 and 165874/2014-7.

References

[1] M. A. Simplicio, M. A. Santos, R. R. Leal, M. A. Gomes, and W. A. Goya, "SecureTCG: a lightweight cheating-detection protocol for P2P multiplayer online trading card games," *Security and Communication Networks*, vol. 7, no. 12, pp. 2412–2431, 2014.

[2] D. Pittman and C. GauthierDickey, "Match+Guardian: a secure peer-to-peer trading card game protocol," *Multimedia systems*, vol. 19, no. 3, pp. 303–314, 2013.

[3] A. Shamir, R. Rivest, and L. Adleman, "Mental poker," in *The Mathematical Gardner*, D. Klarner, Ed. Springer US, 1981, pp. 37–43. [Online]. Available: http://dx.doi.org/10.1007/978-1-4684-6686-7_5

[4] J. Castellà Roca, F. Sebé Feixas, and J. Domingo-Ferrer, *Contributions to mental poker*. Universitat Autònoma de Barcelona,, 2006.

[5] D. Chaum and T. P. Pedersen, "Transferred cash grows in size," in *Advances in Cryptology (Eurocrypt'92)*. Springer, 1993, pp. 390–407.

[6] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in *Advances in Cryptology (Eurocrypt'05)*. Springer, 2005, pp. 302–321.

[7] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "Compact e-cash and simulatable vrfs revisited," in *Pairing'09*. Springer, 2009, pp. 114–131.

[8] G. Fuchsbauer, D. Pointcheval, and D. Vergnaud, "Transferable constant-size fair e-cash," in *Cryptology and Network Security*. Springer, 2009, pp. 226–247.

[9] National Institute of Standards and Technology, *DRAFT FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. pub-NIST, May 2014. [Online]. Available: http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf

[10] M. Abe, J. Groth, M. Ohkubo, and T. Tango, "Converting cryptographic schemes from symmetric to asymmetric bilinear groups," in *Advances in Cryptology (CRYPTO'14)*. Springer, 2014, pp. 241–260.

[11] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology (Eurocrypt'08)*. Springer, 2008, pp. 415–432.

[12] L. Ballard, M. Green, B. de Medeiros, and F. Monrose, "Correlation-resistant storage," TR-SP-BGMM-050507, Johns Hopkins UDCS, Tech. Rep., 2005.

[13] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE, 1999, pp. 120–130.

[14] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Public Key Cryptography (PKC'05)*. Springer, 2005, pp. 416–431.

[15] D. Chaum, "Blind signatures for untraceable payments," in *Advances in cryptology*. Springer, 1983, pp. 199–203.

[16] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo, "Structure-preserving signatures and commitments to group elements," in *Advances in Cryptology (CRYPTO'10)*. Springer, 2010, pp. 209–236.

[17] M. Izabachène, B. Libert, and D. Vergnaud, "Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes," in *Cryptography and Coding*. Springer, 2011, pp. 431–450.

[18] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé, "A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic," in *Advances in Cryptology (Eurocrypt'14)*. Springer, 2014, pp. 1–16.

[19] S. Canard, A. Gouget, and J. Traoré, "Improvement of efficiency in (unconditional) anonymous transferable e-cash," in *Financial Cryptography and Data Security*. Springer, 2008, pp. 202–214.

[20] D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López, "Faster explicit formulas for computing pairings over ordinary curves," in *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 48–68.

Marcos Vinicius Maciel da Silva received his BSc (2013) in Electrical/Computing Engineering at the Escola Politécnica, Universidade de São Paulo, Brazil. He is currently following his MSc on Electrical/Computing Engineering at the same institution. He has experience in the area o Computer Science, with especial interest in the following topics: cloud computing, zero-knowledge proofs, blind signature and electronic cash protocols.

Marcos Antonio Simplicio Junior is an Assistant Professor in the Department of Computer and Digital Systems Engineering at the Escola Politécnica, Universidade de São Paulo, Brazil. He received his BSc (2006), MSc (2008) and PhD (2010) degrees in Electrical/Computing Engineering at the same institution, and also has a Master degree (2006) in Engineering conferred by the Ecole Centrale Des Arts Et Manufactures (Ecole Centrale Paris), France. His main research interests and the focus of the projects coordinated by him include: (applied) cryptography, including the design and analysis of algorithms and protocols; and network security, in particular solutions tailored for scenarios involving resource-constrained devices (e.g., sensor networks), and distributed systems (e.g., cloud computing and P2P networks).