Fair Exchange of Short Signatures without Trusted Third Party

Philippe Camacho

Dept. of Computer Science, University of Chile,
Blanco Encalada 2120, 4to piso, Santiago, Chile
pcamacho@dcc.uchile.cl

Abstract. We propose a protocol to exchange Boneh-Boyen short signatures in a fair way and without relying on a trusted third party. Our protocol is quite practical and is the first of the sort to the best of our knowledge. Our construction uses a new non-interactive zero-knowledge (NIZK) argument to prove that a commitment is the encryption of a bit vector. We also design a NIZK argument to prove that a commitment to a bit vector $v = (b_1, b_2, ..., b_\kappa)$ is such that $\sum_{i \in [\kappa]} b_i 2^{i-1} = \theta$ where θ is the discrete logarithm of some public value $D = g^\theta$. These arguments may be of independent interest.

Keywords: Fair exchange, short signatures, gradual release of a secret.

1 Introduction

Nowadays it is more and more common to trade digital goods on the web: E-books, software licenses, avatar-games currencies like *Ultima Online*[1] to cite a few. Whether these goods are exchanged on *E-bay* through *Paypal* or bought directly to their provider *Amazon* or *Microsoft*, the transaction to be secure requires a trusted third party (TTP). Though it works quite well in practice, enabling totally distributed and at the same time secure transaction systems is of clear interest: It would avoid some security issues due to the presence of single points of failure, and also allow smoother electronic commercial transactions that would not rely on some intermediary. A lot of these transactions may be captured by the exchange of digital signatures. Suppose for example you want to buy a software license to some independent developer: Indeed exchanging the software license as well as the money transfer (digital check) can be modeled by signed messages. However we face a non-trivial problem. Given that the transaction is made on-line, a malicious participant may fool his counterpart by not sending his signature or sending some garbage information. A protocol that prevents such a behavior from a corrupted party is called *fair*: This means that at the end of the execution of protocol either both parties obtain the signature they expected or none does.

---

[1] http://en.wikipedia.org/wiki/Ultima_Online

There are two main approaches to solve this problem. On the one hand, one can assume that both players interact through a TTP. Though this solution does not fit our goal, it is important to note that an important line of research has focused on designing protocols where the TTP is only required when "something goes wrong". These protocols are said to be *optimistically fair*: See [1,25] and [22] for some recent work.

On the other hand, if no TTP exists and we assume that both participants have exactly the same computational resources, then it is impossible *in general* to achieve *complete fairness* [10]. In [2,14] was proposed a way to relax the notion of fairness in order to overcome Cleve's impossibility result. The idea is to assume that both players have roughly the same amount of time, so we can achieve *partial fairness*. Several secure multi-party computations and specific protocols, like [6,11,12,5,16], were built on top of this security notion. The recurrent idea behind these constructions consists in enabling each player to release their secret bit by bit in alternation. Thus, if a player aborts, the other participant will have "only one bit of disadvantage". Formalizing this idea is not an easy task though, in particular because it is hard to reason on the specific amount of time for the players. This issue was noticed in [19] where authors point out that (1) assigning more time to the honest party in order to allow him recover his value is somehow artificial as it does not depend on the participant himself, and (2) implementing such definitions seems to imply the use of strong assumptions related to the *exact* time required to solve some computational problem.

In this work we propose a new security definition that still captures the intuition of partial fairness for the exchange of digital signatures, but without forcing the participants to have access to almost equal computational resources as proposed in [16]. The idea of our definition is to compare the probabilities of computing valid signatures on the agreed messages at the end of the protocol. More precisely, if the adversary aborts the protocol, the honest participant[2] will compute the expected signature by choosing randomly a value from the space of signature candidates, which is defined by the remaining bits to be obtained. The adversary will keep running its own algorithm and also output a signature candidate. We say the protocol is secure if the probabilities that each participant output a valid signature only differ by a polynomial factor. Note that this definition, like previous ones that circumvent Cleve's impossibility result [10], allows the adversary to get some advantage, but it guarantees that this advantage is polynomially bounded. With that definition in hand we can prove the security of our protocol without having to rely on the strong assumptions mentioned above. Our protocol is designed to exchange short signatures [4] without the presence of a TTP. We use bilinear maps as the underlying signature scheme, and also the idea of releasing gradually each bit of some secret θ that will enable to recover the signature. The security of our construction relies on complexity assumptions for bilinear maps, namely the κ-Strong Diffie-Hellman [4], and the κ-Bilinear

---

[2] Note that we need to consider that at least one participant is honest, as otherwise we cannot really avoid that one of the two adversaries, which are arbitrary polynomial time algorithms, wins.

Diffie-Hellman assumptions [3] and holds in the common reference string model. As we use non-interactive zero-knowledge proofs of knowledge (ZKPoK) in order to make the protocol simpler and more efficient, we require the use of random oracle [15] or some non-black box assumptions [20]. If we like, we can use interactive ZKPoK at a minor expense of round efficiency.

OUR CONTRIBUTIONS

1. We propose a practical protocol for exchanging short signatures [4] without relying on a TTP. To the best of our knowledge this is the first construction that meets such a goal. The number of rounds of our protocol is $\kappa + 1$, where κ is the security parameter. The communication complexity is $16\kappa^2 + 12\kappa$ bits. The protocol requires a linear number of group exponentiations, group multiplications, bilinear map applications, hash computations and also a constant number of group divisions.
2. We introduce a new non-interactive zero-knowledge (NIZK) argument to prove that a commitment is the encryption of a bit vector. This protocol may be of independent interest.
3. We introduce another NIZK argument to prove that a commitment to a bit vector corresponds to the binary decomposition of some value θ which is hidden as the discrete logarithm of some group element. We think this argument may lead to other interesting applications.
4. As stated earlier, we propose a new security definition for partial fairness in the context of the exchange of digital signatures. This definition is simple and avoids the issue of involving the *exact* running time of the participants.

OUR APPROACH. Let $\kappa \in \mathbb{N}$ be the security parameter. Let $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow$ BMGen(1^κ) be the public parameter where $p = |\mathbb{G}| = |\mathbb{G}_T|$ is prime, $\mathbb{G}, \mathbb{G}_T$ are cyclic groups, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is the bilinear map and g is a random generator. Let s be a random element in $\mathbb{Z}_p$, we consider the following common reference string: $(g, g^s, g^{s^2}, ..., g^{s^\kappa}) = (g_0, g_1, g_2, ..., g_\kappa)$. In practice this common reference string can be computed using generic multi-party computation techniques (see [9] for an efficient implementation) so that the secret s is randomly generated and remains unknown to all the participants. Another alternative is to rely on a TTP that would "securely delete" the secret after the generation of the common reference string. Obviously the intention of this work is to avoid the use of a TTP, but note however that even in this case, the TTP would be required only once.

Our construction can be summarized as follows. The prover chooses a secret $\theta \in \mathbb{Z}_p$, then commits each bit of this secret into a Pedersen [28] commitment, where the bit b_i in position i with randomness $r_i \in \mathbb{Z}_p$ will be committed with respect to the base (g, g_i): That is $\mathsf{Commit}(b_i, r_i, i) = g^{r_i} g_i^{b_i}$. Then we use a NIZK argument [3] to prove this commitment really encrypts a bit. The next step is to publish $D = g^\theta$ and show, using another NIZK argument,

---

[3] The reader can refer to the full version of this paper [7] for standard definitions related to NIZK protocols.

that θ, the discrete logarithm of D, is "equivalent" to the bit vector committed in $\boldsymbol{C} = (\mathsf{Commit}(b_i, r_i, i))_{i \in [\kappa]}$. More precisely, the argument proves that $\theta = \sum_{i \in [\kappa]} b_i 2^{i-1}$. Now if we consider some signature σ, the prover will blind it using θ to obtain $\tilde{\sigma} = \sigma^\theta$. Using bilinear maps it is straightforward to verify that $\tilde{\sigma}$ contains a valid signature σ which is blinded in the exponent by θ, the discrete logarithm of D. The other verifications will consist simply in checking the NIZK arguments. Finally, we need to provide zero-knowledge proofs of knowledge for the representation of each bit commitment in order to be able to simulate the execution of the protocol even if the adversary aborts. By releasing each bit in turn, both players will reconstruct their own blinding factor θ and obtain the signature.

RELATED WORK. Among the abundant literature on the topic of gradual release and fair exchange for digital signatures, [12] is probably the work that is the most similar to ours: It describes a practical fair exchange protocol for digital signatures based on gradual release of a secret. The protocol described in [12] works for Rabin, RSA and El Gamal signatures. The number of rounds of the protocol described in [12] is roughly 2κ for RSA and Rabin signatures and κ for El Gamal signatures.

Due to Cleve's impossibility result [10], the question of building complete fair protocols with dishonest majority seemed to be closed. However, Gordon et al. showed that non-trivial functions can be computed fairly in the two-party model [18], and left the question of finding a tight characterization of these functions open. In particular it is not known whether functions with a non-polynomial size domain and that return multiple bits as output (like computing a signature) can be computed fairly in Cleve's setting.

In [19] is proposed a definition for partial fairness that may exhibit some similarities with ours (both definitions involve a $Q(\kappa)$ factor where Q is a polynomial). However our definition and approach differs quite from [19]. First, the setting in [19] is more general than our specific construction to exchange digital signatures. Secondly, in their protocol, the number of rounds is variable and defines the level of fairness, whereas in our construction fairness only depends on the computational power of the participants.

Our NIZK argument to prove that a commitment encrypts a bit vector is inspired by [21,20]. We remark that, though [16] uses the idea of gradual release, the construction proposed is not practical as it requires to code the functionality (signing in our case) as an arithmetic circuit.

ORGANIZATION OF THE PAPER. In Section 2 we introduce notations and recall some definitions and standard techniques we use in this work. In Section 3 we describe the bit vector commitment scheme. The argument for proving the equivalence between a bit vector commitment $(C_i)_{i \in [\kappa]}$ and the discrete logarithm θ of g^θ is introduced in Section 4. The fair exchange protocol is shown in Section 5. We conclude in Section 6.

2   Preliminaries

2.1   Notations

For $m, n \in \mathbb{N}$ with $m < n$, $[m..n]$ means the set of integers $\{m, m+1, ..., n-1, n\}$ and $[n]$ means the set of integers $\{1, ..., n\}$. If $\kappa \in \mathbb{N}$ is the security parameter then 1^κ denotes the unary string with κ ones. We will use p to denote a prime number of κ bits. A function $\nu : \mathbb{N} \to [0, 1]$ is said to be negligible in κ if for every polynomial $q(\cdot)$ there exists κ_0 such that $\forall \kappa > \kappa_0 : \nu(\kappa) < 1/q(\kappa)$. In the following, neg will denote *some* negligible function in κ. An algorithm is called PPT if it is probabilistic and runs in polynomial time in κ. We write $x \xleftarrow{R} X$ to denote an element x chosen uniformly at random from a set X. $x \leftarrow v$ means that the variable x is assigned the value v.

A vector of n components and values v_i is denoted $\boldsymbol{v} = (v_i)_{i \in [n]}$. If the vector contains elements of $\mathbb{Z}_p$ we may also write $B[\cdot] = (B[1], B[2], ..., B[n])$. Let $\theta \in \mathbb{Z}_p$, we denote by $\theta[\cdot]$ the binary decomposition (vector) of θ. That is $\theta[\cdot] = (\theta[1], ..., \theta[\kappa])$ and in particular $\theta = \sum_{i \in [\kappa]} \theta[i] 2^{i-1}$. $P(\cdot)$ will stand for a formal polynomial with coefficients in $\mathbb{Z}_p$, and $P[\cdot]$ for the vector of its coefficients: Thus if $d = deg(P)$ is the degree of polynomial $P(\cdot)$ then we have: $P(X) = \sum_{i \in [d+1]} P[i] X^{i-1}$.

2.2   Non-interactive Zero-Knowledge Proofs of Knowledge

Our protocol for fair exchange uses zero-knowledge proofs of knowledge relative to bit commitments. In order to simplify the description of the fair exchange protocol we will use non-interactive zero-knowledge proofs of knowledge. We note however that interactive ZKPoK would work as well, though adding 2 rounds to our protocol and loosing possibly security guarantees in case the protocol is run in parallel or involves more than 2 players. The most popular way to implement such protocols is by using the Fiat-Shamir heuristic [15], trading non-interaction for a security proof relying on the random oracle model. We mention that our scheme could also be adapted to fit Groth's short non-interactive argument proof system [20]. In this case the security of non-interactive proofs of knowledge would depend on a non-black box assumption and we would get shorter arguments[4].

Let $\mathbb{G}$ be a cyclic group of prime order p where the discrete logarithm is hard. Let $\mathsf{H} : \mathbb{G} \to \mathbb{Z}_p$ be a randomly chosen function from a CRHF. Let g, h be two random generators of $\mathbb{G}$ such that the discrete logarithm of h in base g is unknown.

We will need a ZKPoK of the discrete logarithm θ of some public value $D = g^\theta$. Following the notation of [8], we have that $PK\{\theta : g^\theta\} = (c = \mathsf{H}(g^r), z = r - c\theta)$ where $r \xleftarrow{R} \mathbb{Z}_p$. The verifier checks that $c = \mathsf{H}(D^c g^z)$. We will also use the following ZKPoK that convinces a verifier that the prover knows the representation of a commitment $C = g^\alpha h^\beta$ in base (g, h) where $\alpha, \beta \in \mathbb{Z}_p$.

---

[4] Note however that the common reference string would need to be of quadratic size in the size of the statements.

$PK\{(\alpha, \beta) : C = g^\alpha h^\beta\} = (c = \mathsf{H}(g^{r_1} h^{r_2}), z_1 = r_1 - c\alpha, z_2 = r_2 - c\beta)$ where $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$. The verifier checks that $c = \mathsf{H}(C^c g^{z_1} h^{z_2})$.

2.3 Bilinear Maps

In this paper we consider bilinear maps which are defined as following:

Let $\mathbb{G}, \mathbb{G}_T$, be cyclic groups of prime order p. We consider a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ which is

- *bilinear*: $\forall a, b \in \mathbb{G}, x, y \in \mathbb{Z}_p : e(a^x, b^y) = e(a, b)^{xy}$.
- *non-degenerate*: let g be a generator of $\mathbb{G}$ then $e(g, g)$ also generates $\mathbb{G}_T$.
- *efficiently computable*: There exists a polynomial time algorithm BMGen with parameter 1^κ that outputs $(p, \hat{\mathbb{G}}, \hat{\mathbb{G}}_T, \hat{e}, g)$ where $\hat{\mathbb{G}}, \hat{\mathbb{G}}_T$ is the representation of the corresponding groups of size p (p being a prime number of κ bits), g is a generator of $\mathbb{G}$, and $\hat{e}$ is an efficient algorithm to compute the map. For the sake of simplicity, we will not distinguish between $\mathbb{G}, \mathbb{G}_T, e$, and $\hat{\mathbb{G}}, \hat{\mathbb{G}}_T, \hat{e}$.

2.4 Assumptions

Let $N \in \mathbb{N}$. For the following assumptions, the common public parameter is $\mathsf{PP} = < (p, \mathbb{G}, \mathbb{G}_T, e, g), (g_0, g_1, g_2, \cdots, g_N) >$ where s is chosen randomly in $\mathbb{Z}_p$ and $g_i = g^{s^i}$ for $i \in [0..N]$.

Definition 1. N-***Diffie-Hellman Inversion*** (N-***DHI***) ***assumption***, *[26]. The N-Diffie-Hellman Inversion problem consists in computing $g^{\frac{1}{s}}$ given PP. We say the N-DHI assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N\text{-}DHI}(\mathcal{A}, \kappa, N) = \Pr\left[g^{\frac{1}{s}} \leftarrow \mathcal{A}(1^\kappa, \mathsf{PP})\right] = \mathsf{neg}(\kappa)$$

The bilinear variant of the previous assumption was introduced in [3].

Definition 2. N-***Bilinear Diffie-Hellman Inversion assumption*** (N-***BDHI***). *The N- Bilinear Diffie-Hellman Inversion problem consists in computing $e(g, g)^{\frac{1}{s}}$ given PP. We say the N-BDHI assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N\text{-}BDHI}(\mathcal{A}, \kappa, N) = \Pr\left[e(g, g)^{\frac{1}{s}} \leftarrow \mathcal{A}(1^\kappa, \mathsf{PP})\right] = \mathsf{neg}(\kappa)$$

Definition 3. N-***Strong Diffie-Hellman assumption*** (N-***SDH***), *[4]. The N-Strong Diffie-Hellman (N-SDH) problem consists in computing $(c, g^{\frac{1}{s+c}})$ given PP. We say the N-SDH assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N\text{-}SDH}(\mathcal{A}, \kappa, N) = \Pr\left[(c, g^{\frac{1}{s+c}}) \leftarrow \mathcal{A}(1^\kappa, \mathsf{PP})\right] = \mathsf{neg}(\kappa)$$

As mentioned in [4], the N-SDH assumption is equivalent to the N-DHI assumption when c is fixed. The following assumption can be considered as a particular case of the *poly*-Diffie-Hellman assumption [23], or a generalization of the $N+1$-Exponent assumption introduced in [30].

Definition 4. $N+i$-***Diffie-Hellman Exponent*($N+i$-DHE) *assumption.***
The $N+i$-Diffie-Hellman Exponent problem consists in computing $g^{s^{N+i}}$, for
$1 \leq i \leq N$ *given* PP. *We say the $N+i$-DHE assumption holds if for any PPT*
adversary $\mathcal{A}$ we have

$$Adv^{N+i-DHE}(\mathcal{A}, \kappa, N) = \Pr\left[g^{s^{N+i}} \leftarrow \mathcal{A}(1^{\kappa}, \text{PP})\right] = \text{neg}(\kappa)$$

In [30], the N-DHI assumption was shown to be equivalent to the $N+1$-Exponent
assumption ($N+1$-DHE). We state [5] here the following implication.

Proposition 1. N-$BDHI \Rightarrow N+i$-DHE.

2.5   Digital Signatures

STANDARD DIGITAL SIGNATURES. We denote by SSig = (SKG, SSig, SVf) a
standard signature scheme. A pair of private / public keys (sk, pk) is created
by running SKG(1^{κ}). Given a message $m \in \{0, 1\}^*$, a signature on m under
pk is $\sigma_m = \text{SSig}(sk, m)$. A signature σ on m is deemed valid if and only if
SVf(pk, m, σ) returns valid. Regarding security, we use the standard notion of
existential unforgeability under chosen message attack [17].

BONEH AND BOYEN SIGNATURE SCHEME [4]. We recall here briefly the short
signature scheme [4] introduced by Boneh and Boyen. The setup algorithm
BMGen(1^{κ}) generates the public parameters of the scheme $(p, \mathbb{G}, \mathbb{G}_T, e, g)^6$. The
key generation algorithm SKG(1^{κ}) selects random integers $x, y \xleftarrow{R} \mathbb{Z}_p$ and sets
$u = g^x$ and $v = g^y$. The secret key is $sk = (g, x, y)$ and the public key is
$pk = (g, u, v)$. Given a message m and sk, the signing algorithm SSig(sk, m)
works as follows. It selects $r_{\sigma} \xleftarrow{R} \mathbb{Z}_p$ such that $r_{\sigma} - (x + m)/y \neq 0 \bmod p$ and re-
turn the (randomized) signature $\sigma = (g^{\frac{1}{x+m+yr_{\sigma}}}, r_{\sigma}) = (\sigma', r_{\sigma})$. Finally, in order
to verify a signature σ on message m relative to the public key pk, the algorithm
SVf(pk, m, σ) consists in checking that $e(\sigma', ug^m v^{r_{\sigma}}) = e(g, g)$. The scheme is
secure in the standard model under the N-SDH assumption.

2.6   Simultaneous Hardness of Bits for Discrete Logarithm

Our construction relies on the idea of releasing gradually the bits of $\theta \in \mathbb{Z}_p$,
the discrete logarithm in base g of $D = g^{\theta}$. A problem that could arise in this
situation would be that some θ values are somehow easier to find than others,
especially when some of the bits are released. This might help an adversary to
retrieve θ much faster (by a factor greater than a polynomial) and thus break the
security of our protocol. To overcome this issue we need to introduce the *Simul-*
taneous hardness of bits of the discrete logarithm assumption which states that

---

[5] The proof is very similar to the one introduced in [30] and can be found in the full
    version [7].
[6] We use symmetric bilinear map for the sake of exposition.

a polynomial time adversary cannot distinguish[7] between a random sequence of $l = \kappa - \omega(\log \kappa)$ bits and the first l bits of θ when given $D = g^\theta$.

Definition 5. *(Simultaneous hardness of bits for discrete logarithm) Let $\mathbb{G}$ be a cyclic group of prime order p. We say that the* Simultaneous hardness of bits for discrete logarithm *(SHDL) assumption holds, if for every PPT adversary $\mathcal{A}$ and for any $l = \omega(\log \kappa)$, we have that the following quantity is negligible in κ:*

$$Adv^{SHDL}(\mathcal{A}, \kappa) = |\Pr \left[\begin{array}{c} \theta \xleftarrow{R} \mathbb{Z}_p : \\ 1 \leftarrow \mathcal{A}(g^\theta, \theta[1..\kappa - l]) \end{array} \right] - \Pr \left[\begin{array}{c} \theta, \alpha \xleftarrow{R} \mathbb{Z}_p : \\ 1 \leftarrow \mathcal{A}(g^\theta, \alpha[1..\kappa - l]) \end{array} \right]|$$

where the probability is taken over the random choices of $\mathcal{A}$.

Schnorr [29] showed that the $SHDL$ holds in the generic group model by computing the following upper bound on the advantage of the adversary:

$$Adv^{SHDL}(\mathcal{A}, \kappa) = O(\kappa(\kappa - l)\sqrt{t}(\frac{2^{\kappa-l}}{2^\kappa})^{1/4})$$

where t is the number of generic group operations of the adversary. Thus, if we set $l = \omega(\log \kappa)$, we obtain that $Adv^{SHDL}(\mathcal{A}, \kappa) = O(\kappa(\kappa - \omega(\log \kappa)) \sqrt{t}(2^{-\omega(\log \kappa)})^{1/4})$, which is negligible.

The recent work [13] by Duc and Jetchev suggests that results applying to groups of integers modulo a safe prime [27,24] can be extended to elliptic curves so to reduce the SHDL assumption to more standard ones.

3   A New Argument to Prove a Commitment Encrypts a Bit

In this section we describe a commitment scheme to encrypt a vector of values in $\mathbb{Z}_p$ and then provide a NIZK proof that each component of this vector is a bit. Our technique borrows from [21] in the sense we use the idea that if the value b encrypted is a bit then $b(b - 1)$ must be equal to 0, and also from [20] by implementing a basic form of the restriction argument.

Our commitment scheme requires to generate a common reference string $\mathsf{CRS} = (g, g^s, g^{s^2}, ..., g^{s^N}) = (g_0, g_1, ..., g_N)$ where $s \xleftarrow{R} \mathbb{Z}_p$ is the trapdoor. To commit a bit b_i in position i using randomness $r_i \in \mathbb{Z}_p$, we compute the following slight variation of the Pedersen commitment $\mathsf{Commit}(b_i, r_i, i) = C_i = g^{r_i} g_i^{b_i}$. The commitment to the vector $\boldsymbol{B} = (b_1, b_2, ..., b_N)$ using the randomness $\boldsymbol{r} = (r_i)_{i \in [N]}$ will simply be the vector formed by the commitments for each bit in position i: $\boldsymbol{C} = (C_i)_{i \in [N]}$. Abusing a bit our notation, we will write $\boldsymbol{C} = \mathsf{Commit}(\boldsymbol{B}, \boldsymbol{r})$.

---

[7] Note that a PPT adversary can easily distinguish both bit strings if $l = \kappa - O(\log \kappa)$ by performing a brute force attack on the remaining bits as $2^{O(\log \kappa)}$ is a polynomial in κ.

We still need a NIZK that each commitment C_i is the encryption of a bit. The prover proceeds as follows: He computes the "translation" of the commitment by $N - i$ positions to the right, by providing the value $A_i = g_{N-i}^{r_i} g_N^{b_i}$. If we compute $e(A_i, C_i g^{-1})$ and try to express this quantity as $e(B_i, g)$, we realize by simple inspection (see correctness proof of Theorem 1) that a factor $g_{N+i}^{b_i(b_i-1)}$ will appear. Obviously the prover does not know g_{N+i} so in case $b_i \notin \{0, 1\}$ he will not be able to provide the second part of the proof, B_i. If b_i is indeed a bit then the prover will compute the proof $\pi_i = (A_i, B_i)$ in order to convince the verifier that C_i is the encryption of a bit relative to position i. The proofs for the following proposition and theorem can be found in the full version [7].

Proposition 2. *The vector commitment scheme described above is perfectly hiding and computationally binding under the N-BDHI assumption.*

Common reference string: Input $(1^\kappa, N)$

1. $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$
2. $s \overset{R}{\leftarrow} \mathbb{Z}_p$
3. Return $\mathsf{CRS} = < (p, \mathbb{G}, \mathbb{G}_T, g), (g_0, g_1, g_2, ..., g_N) >$ where for all $i \in [0..N] : g_i = g^{s^i}$.

Statement: The statement is formed by a vector of elements of $\mathbb{G}$: $(C_1, C_2, ..., C_N)$. The claim is that for each $i \in [N]$ there exists r_i, b_i such that $C_i = g^{r_i} g_i^{b_i}$ where $b_i \in \{0, 1\}$.

Proof: Input $(\mathsf{CRS}, \boldsymbol{B}, \boldsymbol{r})$

1. Check that $\boldsymbol{B} = (b_1, ..., b_N) \in \{0, 1\}^N$. Return $\bot$ if this is not the case.
2. Check that $\boldsymbol{r} = (r_1, ..., r_N) \in \mathbb{Z}_p^N$. Return $\bot$ if this is not the case.
3. For each $i \in [N]$ compute an argument π_i that C_i is the commitment to a bit in base g_i: $\pi_i = (A_i, B_i)$ where $A_i = C_i^{s^{N-i}}$ and B_i is such that $e(A_i, C_i g_i^{-1}) = e(B_i, g)$.
4. Return $\pi = (\pi_i)_{i \in [N]}$.

Verification: Input $(\mathsf{CRS}, \boldsymbol{C}, \pi)$

1. Parse $\boldsymbol{C}$ as $(C_i)_{i \in [N]}$. Check that $\boldsymbol{C} \in \mathbb{G}^N$.
2. Parse π as $((A_i, B_i))_{i \in [N]}$. Check that $\pi \in (\mathbb{G} \times \mathbb{G})^N$.
3. For each $i \in [N]$ check that:
   (a) $e(C_i, g_{N-i}) = e(A_i, g)$.
   (b) $e(A_i, C_i g_i^{-1}) = e(B_i, g)$.
4. Return valid if and only if all check pass, otherwise return $\bot$.

Fig. 1. NIZK proof of a commitment being the encryption of a binary vector

Theorem 1. *The protocol of Fig. 1 is a NIZK proof that the statement $\boldsymbol{C} = (C_i)_{i \in [N]}$ is such that for every $i \in [N]$ there exists $(r_i, b_i) \in (\mathbb{Z}_p \times \{0, 1\})$ with $C_i = g^{r_i} g_i^{b_i}$. The NIZK proof has perfect completeness, perfect zero-knowledge and computational soundness under the N-BDHI assumption.*

4   Base Equivalence Argument

Let $\theta \xleftarrow{R} \mathbb{Z}_p$. Consider the commitment to the bit vector $\boldsymbol{C} = (C_i)_{i \in [\kappa]} = (g^{r_i} g_i^{\theta[i]})_{i \in [\kappa]}$ where $r_i \in \mathbb{Z}_p$ for each $i \in [\kappa]$ and also $D = g^\theta$. In this section we introduce a NIZK proof to show that indeed each bit commitment in position i, C_i, encrypts the ith bit of θ, which is hidden as the discrete logarithm of D. This argument will allow us to blind the signature with some random factor θ (in the exponent) and then reveal each bit of this exponent gradually without leaking any additional information. The idea is the following. Given $\theta \in \mathbb{Z}_p$ and $\boldsymbol{C} = (g^{r_i} g_i^{\theta[i]})_{i \in [\kappa]}$, the prover proceeds in two steps. First he computes $D' = \frac{\prod_{i \in [\kappa]} g^{r_i} g_i^{\theta[i]}}{g^r}$ where $r = \sum_{i \in [\kappa]} r_i$. Here the prover computes some compressed representation of the bit vector commitment and removes the randomness. Observe however that as θ is uniformly random, thus so is D'. The prover will need to convince the verifier that r is indeed the accumulated randomness of the bit vector commitment. To do so he computes $U = D'^{\frac{1}{s}} = (\prod_{i \in [\kappa]} g_i^{\theta[i]})^{\frac{1}{s}} = \prod_{i \in [\kappa]} g_{i-1}^{\theta[i]}$ where we recall that $g_0 = g$. Observe that this value can be computed without knowing s. In order to verify this proof, the verifier will check that $e(\frac{\prod_{i \in [\kappa]} C_i}{g^r}, g) = e(U, g_1)$. Intuitively, once the randomness of the bit vector is removed one can move the vector to the left by one position. If r would not be equal to $\sum_{i \in [\kappa]} r_i$, this would not be possible without breaking some assumption. The second step consists in checking that the condensed bit vector commitment $U = \prod_{i \in [\kappa]} g_{i-1}^{\theta[i]}$ is "equivalent" to the simple commitment g^θ. This is done by noting that $U = \prod_{i \in [\kappa]} g_{i-1}^{\theta[i]} = g^{P(s)}$ where $P(\cdot)$ is the polynomial $P(X) = \sum_{i \in [\kappa]} \theta[i] X^{i-1}$. This means in particular that $P(2) = \sum_{i \in [\kappa]} \theta[i] 2^{i-1} = \theta$. Thus, we need to prove that $P(s) - P(2) = P(s) - \theta$ is divisible by $s - 2$. The prover can compute the coefficients of the formal polynomial $W(\cdot)$ such that $P(X) - P(2) = W(X)(X - 2)$, then using the common reference string CRS the prover obtains $V = g^{W(s)}$. Verifying the "base equivalence" statement consists in checking that $e(\frac{U}{D}, g) = e(V, g_1 g^{-2}) = e(V, g^{s-2})$. This means that indeed $\theta = P(2)$ and thus the coefficients of $P(\cdot)$ correspond to the binary decomposition of θ. The full protocol is detailed in Fig. 2 and the proof of the theorem is available in the full version [7].

Theorem 2. *The protocol in Fig. 2 is a NIZK proof that the bits of the discrete logarithm of D correspond to the bit vector committed in $(C_i)_{i \in [\kappa]}$. The NIZK proof has perfect completeness, perfect zero-knowledge and computational soundness under the κ-SDH assumption.*

5   Fair Exchange of Short Signatures without TTP

Our fair exchange protocol for digital signatures works as follows. At the beginning a common reference string CRS is generated. Then each participant runs
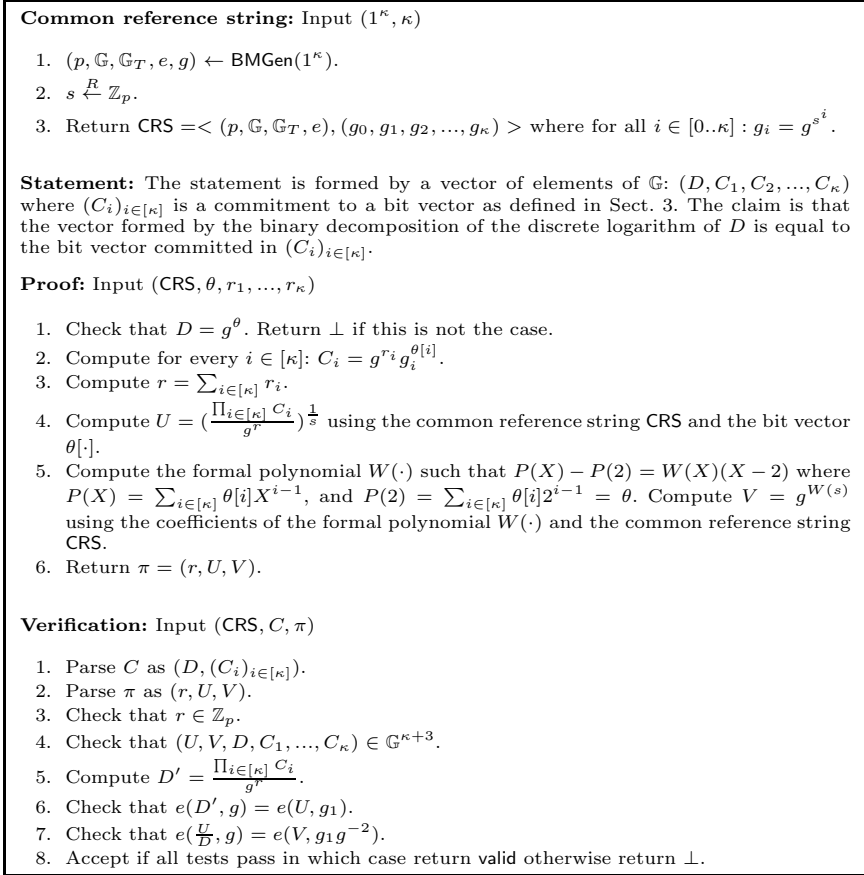

Common reference string: Input $(1^\kappa, \kappa)$

1. $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$.
2. $s \xleftarrow{R} \mathbb{Z}_p$.
3. Return $\mathsf{CRS} = <(p, \mathbb{G}, \mathbb{G}_T, e), (g_0, g_1, g_2, ..., g_\kappa)>$ where for all $i \in [0..\kappa] : g_i = g^{s^i}$.

Statement: The statement is formed by a vector of elements of $\mathbb{G}$: $(D, C_1, C_2, ..., C_\kappa)$ where $(C_i)_{i \in [\kappa]}$ is a commitment to a bit vector as defined in Sect. 3. The claim is that the vector formed by the binary decomposition of the discrete logarithm of D is equal to the bit vector committed in $(C_i)_{i \in [\kappa]}$.

Proof: Input $(\mathsf{CRS}, \theta, r_1, ..., r_\kappa)$

1. Check that $D = g^\theta$. Return $\bot$ if this is not the case.
2. Compute for every $i \in [\kappa]$: $C_i = g^{r_i} g_i^{\theta[i]}$.
3. Compute $r = \sum_{i \in [\kappa]} r_i$.
4. Compute $U = (\frac{\prod_{i \in [\kappa]} C_i}{g^r})^{\frac{1}{s}}$ using the common reference string CRS and the bit vector $\theta[\cdot]$.
5. Compute the formal polynomial $W(\cdot)$ such that $P(X) - P(2) = W(X)(X - 2)$ where $P(X) = \sum_{i \in [\kappa]} \theta[i] X^{i-1}$, and $P(2) = \sum_{i \in [\kappa]} \theta[i] 2^{i-1} = \theta$. Compute $V = g^{W(s)}$ using the coefficients of the formal polynomial $W(\cdot)$ and the common reference string CRS.
6. Return $\pi = (r, U, V)$.

Verification: Input (CRS, C, π)

1. Parse C as $(D, (C_i)_{i \in [\kappa]})$.
2. Parse π as (r, U, V).
3. Check that $r \in \mathbb{Z}_p$.
4. Check that $(U, V, D, C_1, ..., C_\kappa) \in \mathbb{G}^{\kappa+3}$.
5. Compute $D' = \frac{\prod_{i \in [\kappa]} C_i}{g^r}$.
6. Check that $e(D', g) = e(U, g_1)$.
7. Check that $e(\frac{U}{D}, g) = e(V, g_1 g^{-2})$.
8. Accept if all tests pass in which case return valid otherwise return $\bot$.

Fig. 2. NIZK proof that a basic commitment is equivalent to a bit vector commitment

$\mathsf{FEKeyGen}(1^\kappa)$ to obtain a pair of (public/private) keys (pk, sk) for the signing algorithm. At this point each participant executing $\mathsf{EncSigGen}(\mathsf{CRS}, sk, m)$ will compute an encrypted signature γ for the message m, using the signature σ_m blinded with some factor θ. This value γ will also contain the proofs that relate the signature σ_m with some bit vector commitment to θ.

The rest is straightforward: Each participant sends the encrypted signature. If all the verifications pass, the first participant $\mathcal{P}_A$ will ask to $\mathcal{P}_B$ to open the commitment of the first bit of θ_A. If the opening is successful, $\mathcal{P}_B$ will do the same for its own blinding factor θ_B. The process is repeated for each bit until all the bits of the blinding factors are recovered. Finally, each player can compute the signature by "canceling out" the blinding factor θ. The abstract syntax of the protocol is described in Fig. 3.

We describe now more in detail how the encrypted signature is constructed, which is the core of our construction. The encrypted signature contains:

1. A commitment $\boldsymbol{C}$ to the bit string formed by the bits of θ as described in Section 3.
2. $\tilde{\sigma}$, the signature of the message m blinded by θ.
3. Proofs to guarantee that the bit vector commitment encrypts the binary decomposition of the blinding factor θ.
4. A proof in order to convince the verifier that γ is the encryption of σ_m under some blinding factor θ which is hidden in the basic commitment g^{θ}.
5. A proof of knowledge of the discrete logarithm of D and a proof of knowledge of the representation of each bit commitment of the vector $\boldsymbol{C}$. These proofs of knowledge will allow us to keep simulating the adversary despite it aborts.

A detailed description of the concrete protocol is given in Fig. 4.
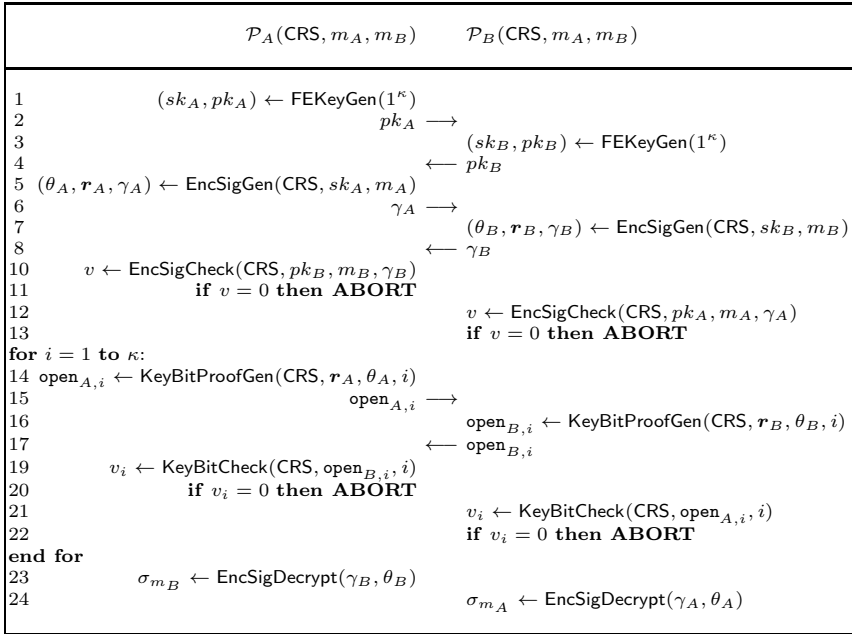

$$\mathcal{P}_A(\mathsf{CRS}, m_A, m_B) \qquad \mathcal{P}_B(\mathsf{CRS}, m_A, m_B)$$

```
1              (sk_A, pk_A) ← FEKeyGen(1^κ)
2                          pk_A  ⟶
3                                          (sk_B, pk_B) ← FEKeyGen(1^κ)
4                               ⟵  pk_B
5   (θ_A, r_A, γ_A) ← EncSigGen(CRS, sk_A, m_A)
6                          γ_A  ⟶
7                                          (θ_B, r_B, γ_B) ← EncSigGen(CRS, sk_B, m_B)
8                               ⟵  γ_B
10      v ← EncSigCheck(CRS, pk_B, m_B, γ_B)
11               if v = 0 then ABORT
12                                          v ← EncSigCheck(CRS, pk_A, m_A, γ_A)
13                                          if v = 0 then ABORT
for i = 1 to κ:
14  open_{A,i} ← KeyBitProofGen(CRS, r_A, θ_A, i)
15                      open_{A,i}  ⟶
16                                          open_{B,i} ← KeyBitProofGen(CRS, r_B, θ_B, i)
17                               ⟵  open_{B,i}
19      v_i ← KeyBitCheck(CRS, open_{B,i}, i)
20               if v_i = 0 then ABORT
21                                          v_i ← KeyBitCheck(CRS, open_{A,i}, i)
22                                          if v_i = 0 then ABORT
end for
23      σ_{m_B} ← EncSigDecrypt(γ_B, θ_B)
24                                          σ_{m_A} ← EncSigDecrypt(γ_A, θ_A)
```

Fig. 3. Abstract fair exchange protocol

We say that the protocol is *perfectly complete*[8] if, and only if, both players $\mathcal{P}_A$ and $\mathcal{P}_B$ that follow the protocol obtain respectively $\sigma_A = \mathtt{SSig}(sk_B, m_B)$, the signature of message m_B and $\sigma_B = \mathtt{SSig}(sk_A, m_A)$, the signature of message m_A, with probability 1.

[8] Here *complete* does not refer to fairness.

We say that the protocol is *(partially) fair* if, at the end of the execution of the protocol (be it normal or anticipated by the abortion of the adversary), the probability of both players to recover their corresponding signature differs at most by a polynomial factor in the security parameter κ. As mentioned in the introduction, the advantage of this approach is that it avoids trying to compare the *exact* running time of the participants and thus allows to capture in a simple, but precise manner, the intuition of *partial fairness*.

Definition 6. *(Partial fairness) We define the* partial fairness *of the protocol through the following experiment: The adversary $\mathcal{A}$ plays the role of the corrupted player say w.l.o.g. $\mathcal{P}_A$. Thus, $\mathcal{P}_B$ is honest and follows the protocol. $\mathcal{O}_{\mathtt{SSig}}(\cdot)$ is the signing oracle for the signature scheme $\mathtt{SSig}$ relative to the public key pk_B of $\mathcal{P}_B$.*

1. *$\mathcal{A}$ asks for signature computations for arbitrary messages to $\mathcal{O}_{\mathtt{SSig}}(\cdot)$.*
2. *$\mathcal{A}$ chooses the messages m_A and m_B on which the fair exchange protocol will be run, with the restriction that m_B must not have been requested before to $\mathcal{O}_{\mathtt{SSig}}(\cdot)$.*
   $\mathcal{A}$ computes also its public key pk_A and sends it to $\mathcal{P}_B$.
3. *$\mathcal{A}$ then interacts in arbitrary way with $\mathcal{P}_B$.*
4. *If $\mathcal{A}$ has aborted before ending the protocol, then let $\theta_A^*[1..i]$ $(0 \leq i \leq \kappa)$ be the partial blinding obtained by $\mathcal{P}_B$. At this point we assume that $\mathcal{P}_B$ will try to compute $\mathtt{SSig}(sk_A, m_A)$ by choosing at random some element in the remaining space of size $2^{\kappa-i}$. We call this tentative signature σ_B.*
5. *$\mathcal{A}$ keeps running its own algorithm and finally outputs a tentative signature σ_A on m_B relative to public key pk_B.*

The protocol is said to be partially fair *if and only if there exists some polynomial $Q(\cdot)$ such that*

$$\frac{\Pr\left[\,\mathtt{SVf}(pk_B, m_B, \sigma_A) = \mathtt{valid}\,\right]}{\Pr\left[\,\mathtt{SVf}(pk_A, m_A, \sigma_B) = \mathtt{valid}\,\right]} \leq Q(\kappa)$$

where the probability is taken over the random choices of $\mathcal{A}$ and $\mathcal{P}_B$.

As the signature scheme presented in [4] is secure under the κ-SDH assumption, we have the following result (see the full version [7] for the proof).

Theorem 3. *The protocol described in Fig. 4 is complete. Moreover if the κ-SDH assumption, the κ-BDHI assumption and the SHDL assumption hold, and a securely precomputed common reference string is available, then it is secure in the random oracle model according to definition 6.*

FESetup(1^κ)

1. $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$

2. $s \xleftarrow{R} \mathbb{Z}_p$

3. Return CRS $=< (p, \mathbb{G}, \mathbb{G}_T, e, g), (g_0, g_1, g_2, ..., g_\kappa) >$ where for all $i \in [0..\kappa] : g_i = g^{s^i}$.

FEKeyGen(1^κ)

1. $(sk, pk) \leftarrow \mathsf{SKG}(1^\kappa)$ where $sk = (g, x, y)$ and $pk = (g, u, v)$ with $u = g^x$ and $v = g^y$, like described in section 2.5.
2. Return (sk, pk).

EncSigGen(CRS, sk, m)

1. Compute $\theta \xleftarrow{R} \mathbb{Z}_p$.
2. Compute $D = g^\theta$.
3. Compute $\boldsymbol{C} = (C_i)_{i \in [\kappa]} = (g^{r_i} g_i^{\theta[i]})_{i \in [\kappa]}$.
4. Compute π_1 that shows that $\boldsymbol{C}$ is the encryption of a binary vector as described in figure 1.
5. Compute π_2 that shows that $\boldsymbol{C}$ is the encryption of the bits of the binary decomposition of the blinding factor θ as described in figure 2.
6. Compute $PK_\theta = PK\{\theta : g^\theta\}$ as described in section 2.2.
7. Compute $\boldsymbol{PK}$, a vector where each component at position i is ZKPoK for the representation of C_i in base (g, g_i). $\boldsymbol{PK} = (PK\{(r_i, \theta[i]) : g^{r_i} g_i^{\theta[i]}\})_{i \in [\kappa]}$ as described in section 2.2.
8. Parse sk as (g, x, y).
9. Set $r_\sigma \xleftarrow{R} \mathbb{Z}_p$.
10. Compute $\sigma = (\sigma', r_\sigma) \leftarrow \mathsf{SSig}(sk, m)$ where $\sigma' = g^{\frac{1}{x+m+yr_\sigma}}$.
11. Set $\tilde{\sigma} \leftarrow (\sigma'^\theta = g^{\frac{\theta}{x+m+yr_\sigma}}, r_\sigma) = (\tilde{\sigma}', r_\sigma)$.
12. Set $\gamma \leftarrow (D, \boldsymbol{C}, \pi_1, \pi_2, PK_\theta, \boldsymbol{PK}, \tilde{\sigma})$.
13. Return $(\theta, \boldsymbol{r}, \gamma)$, where $\boldsymbol{r} = (r_i)_{i \in [\kappa]}$ is the randomness vector of the commitment $\boldsymbol{C}$.

EncSigCheck(CRS, pk, m, γ)

1. Parse γ as $\gamma = (D, \boldsymbol{C}, \pi_1, \pi_2, PK_\theta, \boldsymbol{PK}, \tilde{\sigma})$.
2. Check π_1 as described in figure 1.
3. Check π_2 as described in figure 2.
4. Check PK_θ using D and PK_θ as described in section 2.2.
5. Check the zero-knowledge proof of knowledge $\boldsymbol{PK}$ using $\boldsymbol{C}$ and $\boldsymbol{PK}$ as described in section 2.2.
6. Parse pk as $pk = (g, u, v)$.
7. Check that $e(\tilde{\sigma}, ug^m v_\sigma^r) = e(D, g)$.
8. Return valid if all tests pass, $\perp$ otherwise.

KeyBitProofGen(CRS, $\boldsymbol{r}, \theta, i$)

1. Opens the ith commitment of $\boldsymbol{C}$, that is $(\theta[i], r_i)$ such that $C_i = g^{r_i} g_i^{\theta[i]}$.
2. Return open $\leftarrow (\theta[i], r_i)$.

KeyBitCheck(CRS, open, i)

1. Parse open as open $= (b, r)$
2. Check that $C_i = g^r g_i^b$ and $b \in \{0, 1\}$.

EncSigDecrypt(γ, θ)

1. Parse γ as $\gamma = (D, \boldsymbol{C}, \pi, PK_\theta, \boldsymbol{PK}, \tilde{\sigma})$.
2. Parse $\tilde{\sigma}$ as $\tilde{\sigma} = (\tilde{\sigma}', r_\sigma)$.
3. Compute $\sigma' = \tilde{\sigma}'^{\frac{1}{\theta}}$.
4. Return $\sigma = (\sigma', r_\sigma)$.

Fig. 4. Implementation of the fair exchange protocol

6   Conclusion and Future Work

In this work we introduced a practical protocol to exchange short signatures [4] fairly without relying on a TTP. It seems our approach can be applicable to other signature schemes or more generally to the exchange of values which are computed from a secret and are publicly verifiable using bilinear maps. Thus, our techniques might be extended in order to obtain a general framework to build practical fair protocols involving bilinear maps.

References

1. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: CCS, pp. 7–17. ACM Press (April 1997)
2. Blum, M.: How to exchange (secret) keys. ACM Transactions on Computer Systems 1(2), 175–193 (1983)
3. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology 21(2), 149–177 (2008)
5. Boneh, D., Naor, M.: Timed Commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
6. Brickell, E.F., Chaum, D., Damgård, I.B., van de Graaf, J.: Gradual and Verifiable Release of a Secret. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 156–166. Springer, Heidelberg (1988)
7. Camacho, P.: Fair Exchange of Short Signatures Without Trusted Third Party (2012), `http://eprint.iacr.org/2012/288`
8. Camenisch, J., Stadler, M.: Proof Systems for General Statements about Discrete Logarithms (1997), `ftp://ftp.inf.ethz.ch/pub/crypto/publications/CamSta97b.ps`
9. Choi, S.G., Hwang, K.-W., Katz, J., Malkin, T., Rubenstein, D.: Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 416–432. Springer, Heidelberg (2012)
10. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: STOC, pp. 364–369. ACM Press (November 1986)
11. Cleve, R.: Controlled Gradual Disclosure Schemes for Random Bits and Their Applications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 573–588. Springer, Heidelberg (1990)
12. Damgård, I.: Practical and Provably Secure Release of a Secret and Exchange of Signatures. Journal of Cryptology 8(4), 201–222 (1995)
13. Duc, A., Jetchev, D.: Hardness of Computing Individual Bits for One-Way Functions on Elliptic Curves. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 832–849. Springer, Heidelberg (2012)
14. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM 28(6), 637–647 (1985)

15. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
16. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource Fairness and Composability of Cryptographic Protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer, Heidelberg (2006)
17. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281 (1988)
18. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete Fairness in Secure Two-Party Computation. Journal of the ACM 58(6), 1–37 (2011)
19. Gordon, S.D., Katz, J.: Partial Fairness in Secure Two-Party Computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 157–176. Springer, Heidelberg (2010)
20. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
21. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive Zero Knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
22. Huang, Q., Wong, D.S., Susilo, W.: The Construction of Ambiguous Optimistic Fair Exchange from Designated Confirmer Signature without Random Oracles. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 120–137. Springer, Heidelberg (2012)
23. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010)
24. MacKenzie, P.D., Patel, S.: Hard Bits of the Discrete Log with Applications to Password Authentication. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 209–226. Springer, Heidelberg (2005)
25. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: PODC, pp. 12–19. ACM Press, New York (2003)
26. Mitsunari, S., Sakai, R., Kasahara, M.: A New Traitor Tracing. In: EICE, vol. E 85-A, pp. 481–484 (2002)
27. Patel, S., Sundaram, G.S.: An Efficient Discrete Log Pseudo Random Generator. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 304–317. Springer, Heidelberg (1998)
28. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
29. Schnorr, C.P.: Security of Almost ALL Discrete Log Bits. Electronic Colloquium on Computational Complexity (1998)
30. Zhang, F., Safavi-Naini, R., Susilo, W.: An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)