

# Fair Exchange of Short Signatures without Trusted Third Party

Philippe Camacho  
Dept. of Computer Science, University of Chile,  
Blanco Encalada 2120, 3er piso, Santiago, Chile.  
pcamacho@dcc.uchile.cl

May 22, 2012

## Abstract

We propose a protocol to exchange Boneh-Boyen short signatures in a fair way, 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  $v = (b_1, b_2, \dots, b_\kappa)$  is such that  $\sum_{i \in [\kappa]} b_i 2^{i-1} = \theta$  where  $\theta$  is the discrete logarithm of some public value  $D = g^\theta$ . These arguments may be of independent interest.

**Key words:** 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*<sup>1</sup> 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 modelled by signed messages. However we face a non-trivial problem. Given that the transaction is made online a malicious participant may fool his counterpart by not sending his signature or sending some garbage information. A protocol that prevent such a behaviour 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.

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Ultima\\_Online](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, 19] and [17] for some recent work.

If we assume that both participants have exactly the same computational resources, then it is impossible *in general* to achieve *complete fairness* [7]. In [2, 10] 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 [5, 8, 9, 4, 21, 12], 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 specific amount of time for the players. This issue was noticed in [14] where authors point 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, (2) implementing such definitions seems to imply the use of strong assumptions related to the precise time required to solve some computationally problem, (3) in practice an adversary knowing the exact running time of the honest player may always be able to abort at the precise moment, thus breaking fairness or at least making the recovery of the value (signed message in our case) useless, because obtained too late.

In this work we propose a practical protocol for exchanging short signatures [3] without relying on a TTP. Our protocol uses bilinear maps as the underlying signature scheme. It is also based on the idea of releasing gradually each bit of some secret  $\theta$  that will enable to recover the signature. The security of our scheme relies on complexity assumptions for bilinear maps, namely the  $\kappa$ -Strong Diffie-Hellman [3], and the  $\kappa$  polynomial Diffie-Hellman assumptions [18]. 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 [11] or some non-black box assumptions [15]. If we like we can use interactive ZKPoK at the expense of efficiency and security guarantees.

#### OUR CONTRIBUTIONS.

1. We propose a practical protocol for exchanging short signatures [3] 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  $\kappa + 1$  which is quite efficient as  $\kappa$ , the security parameter is such that  $\kappa \approx 160$ . The communication complexity is  $O(\kappa^2)$  bits. The protocol requires  $O(\kappa)$  group exponentiations, group multiplications, bilinear map applications and also  $O(1)$  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  $\theta$  which is hidden as the discrete logarithm of some group element per participant. We think this argument may lead to other interesting applications.

4. Inspired by [14], we propose a 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 specific running time of the participants.

We finally note that though we use our techniques to the specific scheme of [3], the proposed approach is flexible enough to be applied to other signature constructions or more generally to schemes that involve some value computed from a secret and which is publicly checkable.

**OUR APPROACH.** Our construction uses bilinear maps. Let  $\kappa \in \mathbb{N}$  be the security parameter. Let  $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \text{BMGen}(1^\kappa)$  be the public parameter where  $p = |\mathbb{G}| = |\mathbb{G}_T|$  is prime,  $\mathbb{G}, \mathbb{G}_T$  are cyclic groups,  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  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:  $(g, g^s, g^{s^2}, \dots, g^{s^\kappa}) = (g_0, g_1, g_2, \dots, g_\kappa)$ .

Our construction can be summarized as follows. The prover chooses a secret  $\theta$ , then commits each bit of this secret into a pedersen [20] 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  $\text{Commit}(b_i, r_i, i) = g^{r_i} g_i^{b_i}$ . Then we use a NIZK argument to prove this commitment really encrypts a bit. The next step is to publish  $D = g^\theta$  and show, using another NIZK argument, that  $\theta$  the discrete logarithm of  $D$  is “equivalent” to the bit vector committed in  $\vec{C} = (\text{Commit}(b_i, r_i, i))_{i \in [\kappa]}$ . More precisely the argument proves that  $\theta = \sum_{i \in [\kappa]} b_i 2^{i-1}$ . Now if we consider some signature  $\sigma$ , the prover will blind it using  $\theta$  to obtain  $\tilde{\sigma} = \sigma^\theta$ . Using bilinear maps it is straightforward to verify that  $\tilde{\sigma}$  contains a valid signature  $\sigma$  which is blinded in the exponent by  $\theta$ , 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  $\theta$  and obtain the signature.

**RELATED WORK.** Among the abundant literature on the topic of gradual release and fair exchange for digital signatures, [9] 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 main difference is that our gradual release protocol is designed for bilinear maps schemes, in particular [3], while [9] works for El-Gamal, Rabin or RSA compatible signatures.

Our NIZK argument to prove that a commitment encrypts a bit vector is inspired by [16, 15]. We remark that though [12] 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  $\theta$  of  $g^\theta$  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  $\{m, m+1, \dots, n-1, n\}$  and  $[n]$  means the set of integers  $\{1, \dots, n\}$ . If  $\kappa \in \mathbb{N}$  is the security parameter then  $1^\kappa$  denotes the unary string with  $\kappa$  ones. We will use  $p$  to denote a prime number of  $\kappa$  bits. A function  $\nu : \mathbb{N} \rightarrow [0, 1]$  is said to be negligible in  $\kappa$  if for every polynomial  $p(\cdot)$  there exists  $\kappa_0$  such that  $\forall \kappa > \kappa_0 : \nu(\kappa) < 1/p(\kappa)$ . In the following,  $\text{neg}$  will denote *some* negligible function in  $\kappa$ . An algorithm is called PPT if it is probabilistic and runs in polynomial time in  $\kappa$ . We write  $x \stackrel{R}{\leftarrow} X$  to denote an element  $x$  chosen uniformly at random from a set  $X$ .  $x \leftarrow v$  means that variable  $x$  is assigned value  $v$ .

A vector of  $n$  components and values  $v_i$  is denoted  $\vec{v} = (v_i)_{i \in [n]}$ . If the vector contains elements of  $\mathbb{Z}_p$  we may also write  $B[\cdot] = (B[1], B[2], \dots, B[n])$ .

Let  $\theta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ , we denote by  $\theta[\cdot]$  the binary decomposition (vector) of  $\theta$ . That is  $\theta[\cdot] = (\theta[1], \dots, \theta[\kappa])$  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.

### 2.2 Bilinear Maps

Let  $\mathbb{G}, \mathbb{G}_T$ , be cyclic groups of prime order  $p$ . We consider a map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  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  $\text{BMGen}$  with parameter  $1^\kappa$  that outputs  $(p, \hat{\mathbb{G}}, \hat{\mathbb{G}}_T, \hat{e}, g)$  where  $\hat{\mathbb{G}}, \hat{\mathbb{G}}_T$  refer to the representation of both groups of size  $p$  ( $p$  being a prime number of  $\kappa$  bits),  $g$  is a generator of  $\mathbb{G}$  and  $\hat{e}$  is an efficient algorithm to compute the map. For the sake of simplicity, in the following we will not distinguish between  $\mathbb{G}, \mathbb{G}_T, e$  and  $\hat{\mathbb{G}}, \hat{\mathbb{G}}_T, \hat{e}$ .

The following assumption was introduced in [18].

**Definition 1**  *$\lambda$ -polynomial Diffie-Hellman ( $\lambda$ -polyDH) assumption.* Let  $P = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \text{BMGen}(1^\kappa)$ ,  $s \stackrel{R}{\leftarrow} \mathbb{Z}_p$ , and  $T = (g, g^s, g^{s^2}, \dots, g^{s^\lambda})$  for  $\lambda \in \mathbb{N}$ . The  $\lambda$ -polynomial Diffie-Hellman problem consists in computing  $\langle Q(\text{c}dot), g^{Q(s)} \rangle$ , given  $P$  and  $T$  such that  $2^\kappa > \deg(Q) > \lambda$ . We say the  $\lambda$ -polyDH assumption holds if for any PPT adversary  $\mathcal{A}$  we have:

$$\text{Adv}^{\lambda\text{-polyDH}}(\mathcal{A}, \kappa, \lambda) = \Pr \left[ \langle Q(\text{c}dot), g^{Q(s)} \rangle \leftarrow \mathcal{A}(1^\kappa, P, T) \wedge 2^\kappa > \deg(Q) > \lambda \right] = \text{neg}(\kappa)$$

The following assumption was introduced in [3].

**Definition 2**  *$\lambda$ -Strong Diffie-Hellman ( $\lambda$ -SDH) assumption.* Let  $P = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \text{BMGen}(1^\kappa)$ ,  $s \stackrel{R}{\leftarrow} \mathbb{Z}_p$ , and  $T = (g, g^s, g^{s^2}, \dots, g^{s^\lambda})$  for  $\lambda \in \mathbb{N}$ . The  $\lambda$ -Strong Diffie-Hellman ( $\lambda$ -SDH) problem consists in computing  $(c, W)$ , such that  $W = g^{\frac{1}{s+c}}$ . given  $P$  and  $T$ . We say the  $\lambda$ -SDH assumption holds if for any PPT adversary  $\mathcal{A}$  we have:

$$\text{Adv}^{\lambda\text{-SDH}}(\mathcal{A}, \kappa, \lambda) = \Pr \left[ (c, W) \leftarrow \mathcal{A}(1^\kappa, P, T) \wedge W = g^{\frac{1}{s+c}} \right] = \text{neg}(\kappa)$$

### 2.3 Boneh and Boyen signature scheme [3]

We recall here briefly the short signature scheme [3] introduced by Boneh and Boyen. The setup algorithm  $\text{BGen}(1^\kappa)$  generates the public parameters of the scheme  $(p, \mathbb{G}, \mathbb{G}_T, e, g)^2$ . The key generation algorithm  $\text{SKG}(1^\kappa)$  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  $\text{SSig}(sk, m)$  works as follows. It selects  $r_\sigma \xleftarrow{R} \mathbb{Z}_p$  such that  $r_\sigma - (x + m)/y \neq 0 \pmod p$  and return the (randomized) signature  $\sigma = (g^{\frac{1}{x+m+yr_\sigma}}, r_\sigma) = (\sigma', r_\sigma)$ . Finally in order to verify a signature  $\sigma$  on message  $m$ , relative to the public key  $pk$ , the algorithm  $\text{SVf}(pk, m, \sigma)$  consists in checking that  $e(\sigma', ug^m v^{r_\sigma}) = e(g, g)$ .

Note that the security definition proposed in [3] is slightly stronger compared to the one introduced in [13]: The adversary is allowed to output signatures for messages already queried to the signing oracle but with the restriction that this signature should have different randomness. In other words the scheme is such that a same message can have many valid signatures, and finding new signatures for an already signed message should be hard for an attacker. Finally the scheme is secure in the standard model under the  $\lambda - \text{SDH}$  assumption.

### 2.4 Non-interactive Zero-Knowledge Arguments

We are interested in statements that are efficiently verifiable<sup>3</sup>. Let  $R$  be a NP relation such that  $(C, w) \in R$  means the statement is true and this can be verified with the witness  $w$ . We will consider  $R_\lambda$ , the subset of  $R$  where the statements are of size  $\lambda = \kappa^{O(1)}$ . For relation  $R$  we define a non-interactive argument in the following way. An algorithm  $\mathcal{K}(1^\kappa, \lambda)$  generates the common reference string  $\text{CRS}$ . Then the prover  $\mathcal{P}$  given as input  $(\text{CRS}, C, w)$ , check first that  $(C, w) \in R_\lambda$ . If this is not the case he outputs  $\perp$ . Otherwise he outputs an argument  $\pi$ . The verifier  $\mathcal{V}$  using  $\text{CRS}, C$  and  $\pi$  returns 1 in case it accepts the argument and 0 otherwise.

In our case,  $C$  will be a commitment and  $w$  its opening (the message and the randomness). We will consider non-interactive zero-knowledge (NIZK) argument (proof) systems  $(\mathcal{K}, \mathcal{P}, \mathcal{V})$  for the relation  $R_\lambda$  with the following properties.

**PERFECT COMPLETENESS.** The argument is perfectly complete if a honest prover can convince a honest verifier with probability 1 in case the statement is true. For any PPT adversary  $\mathcal{A}$  we have

$$\Pr \left[ \begin{array}{l} \text{CRS} \leftarrow \mathcal{K}(1^\kappa, \lambda); (C, w) \leftarrow \mathcal{A}(\text{CRS}); \pi \leftarrow \mathcal{P}(\text{CRS}, C, w) : \\ \mathcal{V}(\text{CRS}, C, \pi) = 1 \wedge (C, w) \in R_\lambda \end{array} \right] = 1$$

**COMPUTATIONAL SOUNDNESS.** The argument is said to be sound if no adversary can convince a verifier of a false statement. For any PPT adversary  $\mathcal{A}$  we have

$$\Pr \left[ \begin{array}{l} \text{CRS} \leftarrow \mathcal{K}(1^\kappa, \lambda); (C, \pi) \leftarrow \mathcal{A}(\text{CRS}) : \\ \mathcal{V}(\text{CRS}, C, \pi) = 1 \wedge \nexists w : (C, w) \in R_\lambda \end{array} \right] = \text{neg}(\kappa)$$

**PERFECT WITNESS INDISTINGUISHABILITY.** The argument is said to be perfectly witness-indistinguishable if the verifier does not learn which witness was used by the prover in order to produce the proof.

<sup>2</sup>We use symmetric bilinear map for the sake of simplicity.

<sup>3</sup>We follow the notations of [15].

For all stateful interactive PPT adversaries  $\mathcal{A}$  we have

$$\begin{aligned} & \Pr \left[ \begin{array}{l} \text{CRS} \leftarrow \mathcal{K}(1^\kappa, \lambda); (C, w_1, w_2) \leftarrow \mathcal{A}(\text{CRS}); \pi \leftarrow \mathcal{P}(\text{CRS}, C, w_1) : \\ ((C, w_1), (C, w_2)) \in R_\lambda^2 \wedge \mathcal{A}(\pi) = 1 \end{array} \right] \\ = & \Pr \left[ \begin{array}{l} \text{CRS} \leftarrow \mathcal{K}(1^\kappa, \lambda); (C, w_1, w_2) \leftarrow \mathcal{A}(\text{CRS}); \pi \leftarrow \mathcal{P}(\text{CRS}, C, w_2) : \\ ((C, w_1), (C, w_2)) \in R_\lambda^2 \wedge \mathcal{A}(\pi) = 1 \end{array} \right] \end{aligned}$$

Note that in case there is only one valid witness  $w$  for some statement  $C$ , then the argument becomes trivially perfectly witness-indistinguishable.

**PERFECT ZERO-KNOWLEDGE.** We say an argument is zero-knowledge if the verifier learns nothing but the truth of the statement. To formalize this idea we consider two simulators  $S_1, S_2$  such that  $S_1$  generates the CRS and a trapdoor  $\tau$ . The simulator  $S_2$  uses CRS, the statement  $C$  and the trapdoor  $\tau$  to output a simulated argument  $\pi$ . The argument is said to be perfect zero-knowledge if for any stateful interactive PPT adversary  $\mathcal{A}$  we have

$$\Pr \left[ \begin{array}{l} \text{CRS} \leftarrow \mathcal{K}(1^\kappa, \lambda); \\ (C, w) \leftarrow \mathcal{A}(\text{CRS}); \\ \pi \leftarrow \mathcal{P}(\text{CRS}, C, w) : \\ (C, w) \in R_\lambda \wedge \mathcal{A}(\pi) = 1 \end{array} \right] = \Pr \left[ \begin{array}{l} (\text{CRS}, \tau) \leftarrow S_1(1^\kappa, \lambda); \\ (C, w) \leftarrow \mathcal{A}(\text{CRS}); \\ \pi \leftarrow S_2(\text{CRS}, C, \tau) : \\ (C, w) \in R_\lambda \wedge \mathcal{A}(\pi) = 1 \end{array} \right]$$

## 2.5 Non-interactive Zero-Knowledge Proofs of Knowledge

Our protocol 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. As mentioned in the introduction, we note however that interactive ZKPoK would work as well, though adding 2 rounds to our protocol and losing 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 [11], 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 [15]. 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<sup>4</sup>.

Let  $\mathbb{G}$  be a cyclic group of prime order  $p$  where the discrete logarithm is hard. Let  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$  be a randomly chosen function from a collision-resistant hash function family. Let  $g, h$  be two random generators of  $\mathbb{G}$  such that the discrete logarithm of  $h$  in base  $g$  is unknown.

We will 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  $\alpha, \beta \in \mathbb{Z}_p$ . Following the notation of [6] we have that  $PK\{(\alpha, \beta) : C = g^\alpha h^\beta\} = (c = H(g^{r_1} h^{r_2}), z_1 = r_1 - c\alpha, z_2 = r_2 - c\beta)$  where  $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$ . The verifier checks that  $c = H(C^c g^{z_1} h^{z_2})$ .

## 3 A new bit vector commitment scheme

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

<sup>4</sup>Note however that the common reference string would need to be of quadratic size in the size of the statements.

[16] 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 [15] by implementing a basic form of the restriction argument.

Our commitment scheme requires to generate a common reference string  $\text{CRS} = (g, g^s, g^{s^2}, \dots, g^{s^\lambda}) = (g_0, g_1, \dots, g_\lambda)$  where  $s \xleftarrow{R} \mathbb{Z}_p$ . 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 [20] commitment  $\text{Commit}(b_i, r_i, i) = C_i = g^{r_i} g_i^{b_i}$ . The commitment to the vector  $\vec{B} = (b_1, b_2, \dots, b_\lambda)$  using the randomness  $(r_i)_{i \in [\lambda]}$  will simply be the vector formed by the commitments for each bit in position  $i$ :  $\vec{C} = (C_i)_{i \in [\lambda]}$ .

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  $\lambda - i$  positions to the right, by providing the value  $A_i = g_{\lambda-i}^{r_i} g_\lambda^{b_i}$ . 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  $g^{b_i(b_i-1)s^{\lambda+i}}$  will appear. Obviously the prover does not know  $g_{\lambda+i}$  so in case  $b \notin \{0, 1\}$  he will not be able to provide the second part of the proof,  $B_i$ . In case  $b$  is indeed a bit then the prover will compute the proof  $\pi = (A_i, B_i)$  in order to convince the verifier that  $C_i$  is the encryption of a bit relative to position  $i$ .

**Proposition 1** *The vector commitment scheme described above is perfectly hiding and computationally binding under the  $\lambda - \text{polyDH}$  assumption.*

**Proof.**

We observe that each element in position  $[i \in \lambda]$  of the vector is a Pedersen commitment where  $h = g^{s^i}$  and the discrete logarithm of  $h$  in base  $g$ ,  $s^i$ , is unknown. We can deduce directly that the commitment scheme is perfectly hiding.

Assume an adversary  $\mathcal{A}$  computes  $B[\cdot], B'[\cdot]$  two vectors and  $\vec{r} = (r_1, \dots, r_\lambda), \vec{r}' = (r'_1, \dots, r'_\lambda) \in \mathbb{Z}_p^\lambda$  two randomness vectors such that  $\text{Commit}(\text{CRS}, \vec{B}, \vec{r}) = \text{Commit}(\text{CRS}, \vec{B}', \vec{r}')$  and  $B[j] \neq B'[j]$  for (at least) one  $j \in [\lambda]$ : We obtain the equation  $g^{r_j - r'_j} g^{(B[j] - B'[j])s^j} = 1_G$ . If we set  $X = s^j$ , we can deduce that  $(r_j - r'_j) + (B[j] - B'[j])X = 0$  and then  $X = s^j = \frac{r'_j - r_j}{B[j] - B'[j]} \bmod p$ . Once  $s^j$  is recovered we can compute  $g_\lambda^X = g_{\lambda+j}$  and the  $\lambda - \text{polyDH}$  assumption is broken.

■

**Theorem 1** *The protocol of figure 1 is a NIZK proof that the statement  $C = (C_i)_{i \in [\lambda]}$  is such that for every  $i \in [\lambda]$  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  $\lambda - \text{polyDH}$  assumption.*

See section B for the proof.

## 4 Base equivalence argument

Let  $\theta \xleftarrow{R} \mathbb{Z}_p$ . Consider the commitment to the bit vector  $\vec{C} = (C_i)_{i \in [\kappa]} = (g^{r_i} g_i^{\theta[i]})_{i \in [\kappa]}$  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  $i$  of  $C_i$  encrypts the  $i$ -th bit of  $\theta$ , which is hidden

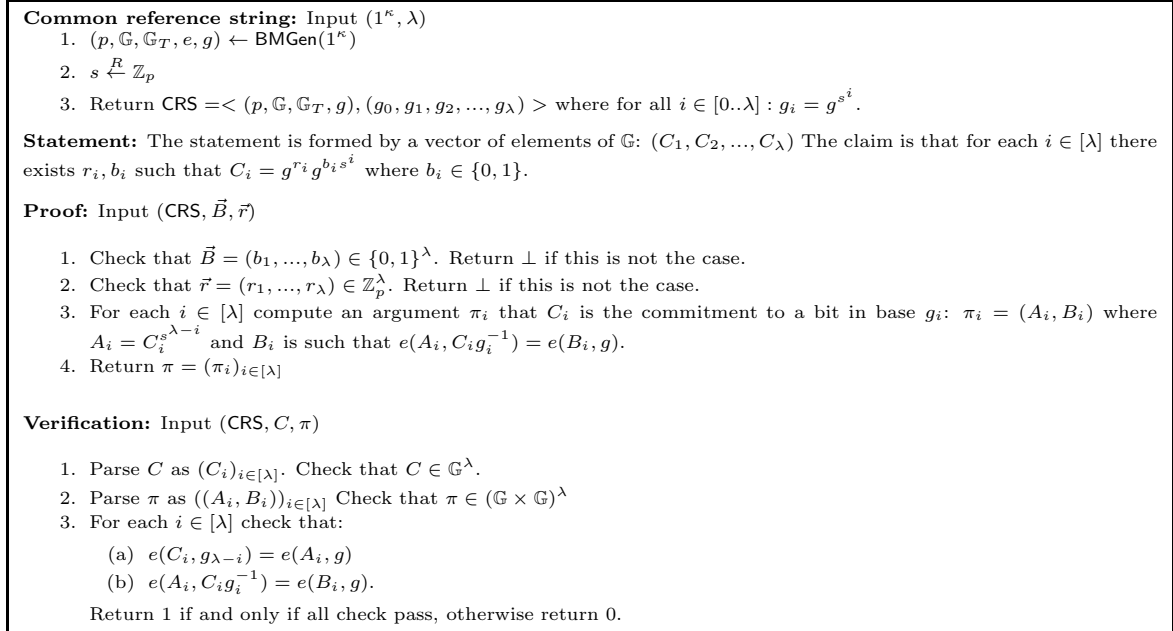


Figure 1: NIZK proof of a commitment being the encryption of a binary vector.

as the discrete logarithm of  $D$ . This argument will allow us to blind the signature with some factor  $\theta$  (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  $\vec{C} = (g^{r_i} g_i^{\theta[i]})_{i \in [\kappa]}$ , 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  $\theta$  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 by convention  $g_0 = g$ . Observe that this value can be computed without knowing  $s$ . In order to verify this proof, the verifier will verify 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  $g^\theta$ . 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  $\text{CRS}$  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  $\theta$ . The full



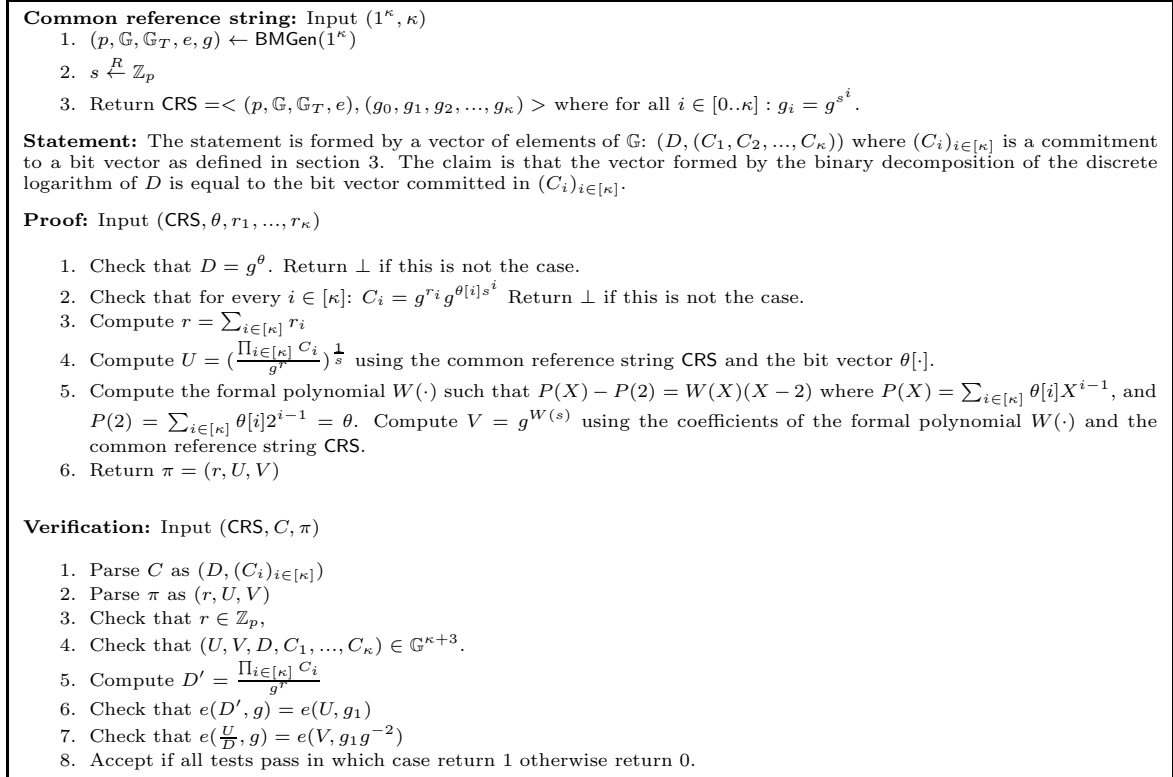


Figure 2: NIZK proof that a basic commitment is equivalent to a bit vector commitment.

protocol is detailed in figure 2.

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

See section C for the proof.

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

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  $\theta$ . If

the opening is successful  $\mathcal{P}_B$  will do the same for its own blinding factor. 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 “cancelling out” the blinding factor  $\theta$ . The abstract syntax of the protocol is described in figure 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  $\vec{C}$  to the bit string formed by the bits of  $\theta$  as described in section 3.
2.  $\tilde{\sigma}$ , the signature of the message  $m$  blinded by  $\theta$ .
3. Proofs to guarantee that the bit vector commitment encrypts the binary decomposition of the blinding factor  $\theta$ .
4. A proof in order to convince the verifier that  $\gamma$  is the encryption of  $\sigma_m$  under some blinding factor  $\theta$  which is hidden in the basic commitment  $g^\theta$ .
5. A proof of knowledge of the representation of each bit commitment of the vector  $\vec{C}$ . These proofs of knowledge will allow us to keep simulating the adversary despite he aborts.

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

	$\mathcal{P}_A(m_A, m_B, \text{CRS})$		$\mathcal{P}_B(m_A, m_B, \text{CRS})$
1	$(pk_A, sk_A) \leftarrow \text{FEKeyGen}(1^\kappa)$		
2	$pk_A$	→	
3			$(pk_B, sk_B) \leftarrow \text{FEKeyGen}(1^\kappa)$
4		←	$pk_B$
5	$(\theta_A, \vec{r}_A, \gamma_A) \leftarrow \text{EncSigGen}(\text{CRS}, sk_A, m_A)$		
6	$\gamma_A$	→	
7			$(\theta_B, \vec{r}_B, \gamma_B) \leftarrow \text{EncSigGen}(\text{CRS}, sk_B, m_B)$
8		←	$\gamma_B$
10	$v \leftarrow \text{EncSigCheck}(\text{CRS}, pk_B, m_B, \gamma_B)$		
11	<b>if</b> $v = 0$ <b>then</b> <b>ABORT</b>		
12			$v \leftarrow \text{EncSigCheck}(\text{CRS}, pk_A, m_A, \gamma_A)$
13			<b>if</b> $v = 0$ <b>then</b> <b>ABORT</b>
14	<b>for</b> $i = 1$ <b>to</b> $\kappa$ :		
15	$\text{open}_{A,i} \leftarrow \text{KeyBitProofGen}(\text{CRS}, \vec{r}_A, \theta_A, i)$		
16	$\text{open}_{A,i}$	→	
17			$\text{open}_{B,i} \leftarrow \text{KeyBitProofGen}(\text{CRS}, \vec{r}_B, \theta_B, i)$
18		←	$\text{open}_{B,i}$
19	$v_i \leftarrow \text{KeyBitCheck}(\text{CRS}, \text{open}_{B,i}, i)$		
20	<b>if</b> $v_i = 0$ <b>then</b> <b>ABORT</b>		
21			$v_i \leftarrow \text{KeyBitCheck}(\text{CRS}, \text{open}_{A,i}, i)$
22			<b>if</b> $v_i = 0$ <b>then</b> <b>ABORT</b>
23	<b>end for</b>		
24	$\sigma_{m_B} \leftarrow \text{EncSigDecrypt}(\gamma_B, \theta_B)$		$\sigma_{m_A} \leftarrow \text{EncSigDecrypt}(\gamma_A, \theta_A)$

Figure 3: Abstract Fair Exchange Protocol.

We say that the protocol is *perfectly complete*<sup>5</sup> if and only if both players  $\mathcal{P}_A$  and  $\mathcal{P}_B$  that follow the protocol obtain respectively  $\sigma_A = \text{SSig}(sk_B, m_B)$ , the signature of message  $m_B$  and  $\sigma_B = \text{SSig}(sk_A, m_A)$ , the signature of message  $m_A$  with probability 1.

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  $\kappa$ . As mentioned in the introduction, the advantage of this approach is that it avoids trying to compare the running time of the participants and thus allows to capture in a simple, but precise manner the intuition of *partial fairness*.

**Definition 3** (*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.  $\mathcal{O}_{\text{SSig}}(\cdot)$  is the signing oracle for the signature scheme  $\text{SSig}$  relative to the public key  $pk_B$  of  $\mathcal{P}_B$ .

1.  $\mathcal{A}$  asks for signature computations for arbitrary messages to  $\mathcal{O}_{\text{SSig}}(\cdot)$ .
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  $\mathcal{O}_{\text{SSig}}(\cdot)$ .  
 $\mathcal{A}$  computes also its public key  $pk_A$  and sends it to  $\mathcal{P}_B$ .
3. Then  $\mathcal{P}_B$  (following the protocol) sends his public key  $pk_B$  (which is the one of  $\mathcal{O}_{\text{SSig}}(\cdot)$ ) to  $\mathcal{A}$ .
4.  $\mathcal{A}$  then interacts in arbitrary way with  $\mathcal{P}_B$ .
5. If  $\mathcal{A}$  has aborted before ending the protocol, then let  $\theta_A^*[1..i]$  ( $1 \leq i \leq \kappa$ ) be the partial blinding obtained by  $\mathcal{P}_B$ . At this point we assume that  $\mathcal{P}_B$  will try to compute  $\text{SSig}(sk_A, m_A)$  by choosing at random some element in the remaining space of size  $2^{\kappa-i}$ . We call this tentative signature  $\sigma_B$ .
6.  $\mathcal{A}$  keeps running his own algorithm and finally outputs a tentative signature  $\sigma_A$  on  $m_B$  relative to public key  $pk_B$ .

The protocol is said to be partially fair if and only if

$$\text{Adv}^{\text{FE}}(\mathcal{A}, \kappa) = \frac{\Pr[\text{SVf}(pk_B, m_B, \sigma_A) = \text{valid}]}{\Pr[\text{SVf}(pk_A, m_A, \sigma_B) = \text{valid}] \cdot 2^\kappa} = \text{neg}(\kappa)$$

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

**Theorem 3** The protocol described in figure 4 is complete. Moreover if the  $\kappa$ -SDH assumption and the  $\kappa$ -polyDH assumptions hold and the underlying signature scheme is secure, it is secure in the random oracle model according to definition 3.

A sketch of the proof is given in section A.

---

<sup>5</sup>Here *complete* does not refer to fairness.

<p><b>FESetup</b>(<math>1^\kappa</math>)</p> <ol style="list-style-type: none"> <li>1. <math>(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \text{BMGen}(1^\kappa)</math></li> <li>2. <math>s \xleftarrow{R} \mathbb{Z}_p</math></li> <li>3. Return CRS = <math>\langle (p, \mathbb{G}, \mathbb{G}_T, e, g), (g_0, g_1, g_2, \dots, g_\kappa) \rangle</math> where for all <math>i \in [0..\kappa]</math> : <math>g_i = g^{s^i}</math>.</li> </ol> <p><b>FEKeyGen</b>(<math>1^\kappa</math>)</p> <ol style="list-style-type: none"> <li>1. <math>(sk, pk) \leftarrow \text{SKG}(1^\kappa)</math> where <math>sk = (g, x, y)</math> and <math>pk = (g, u, v)</math> like described in section 2.3.</li> <li>2. Return <math>(sk, pk)</math>.</li> </ol> <p><b>EncSigGen</b>(CRS, <math>sk, m</math>)</p> <ol style="list-style-type: none"> <li>1. Compute <math>\theta \xleftarrow{R} \mathbb{Z}_p</math>.</li> <li>2. Compute <math>(D, \vec{C}, \vec{r})</math> so that <math>D = g^\theta</math> and <math>\vec{C} = (C_i)_{i \in [\kappa]} = (g^{r_i} g^{\theta[i]s^i})_{i \in [\kappa]}</math></li> <li>3. Compute <math>\pi_1</math> that shows that <math>\vec{C}</math> is the encryption of a binary vector as described in figure 1.</li> <li>4. Compute <math>\pi_2</math> that shows that <math>\vec{C}</math> is the encryption of the bits of the blinding factor <math>\theta</math> as described in figure 2.</li> <li>5. Compute <math>\vec{PK}</math>, a vector where each component at position <math>i</math> is ZKPoK for the representation of <math>C_i</math> in base <math>(g, g_i)</math>.  <math>\vec{PK} = (PK\{(r_i, \theta[i]) : g^{r_i} (g^{s^i})^{\theta[i]}\})_{i \in [\kappa]}</math> as described in 2.5.</li> <li>6. Parse <math>sk</math> as <math>(g, x, y)</math>.</li> <li>7. Set <math>r_\sigma \xleftarrow{R} \mathbb{Z}_p</math>.</li> <li>8. Compute <math>\sigma = (\sigma', r_\sigma) \leftarrow \text{SSig}(sk, m)</math> where <math>\sigma' = g^{\frac{1}{x+m+y r_\sigma}}</math></li> <li>9. Set <math>\vec{\sigma} \leftarrow (\sigma'^\theta = g^{\frac{\theta}{x+m+y r_\sigma}}, r_\sigma) = (\vec{\sigma}', r_\sigma)</math>.</li> <li>10. Set <math>\gamma \leftarrow (D, \vec{C}, \pi_1, \pi_2, \vec{PK}, \vec{\sigma})</math>.</li> <li>11. Return <math>(\theta, \vec{r}, \gamma)</math>.</li> </ol> <p><b>EncSigCheck</b>(CRS, <math>pk, m, \gamma</math>)</p> <ol style="list-style-type: none"> <li>1. Parse <math>\gamma</math> as <math>\gamma = (D, \vec{C}, \pi_1, \pi_2, \vec{PK}, \vec{\sigma})</math></li> <li>2. Check <math>\pi_1</math> as described in figure 1.</li> <li>3. Check <math>\pi_2</math> as described in figure 2.</li> <li>4. Check the zero-knowledge proof of knowledge using <math>\vec{C}</math> and <math>\vec{PK}</math> as described in section 2.5.</li> <li>5. Parse <math>pk</math> as <math>pk = (g, u, v)</math></li> <li>6. Check that <math>e(\vec{\sigma}, u g^m v_\sigma^r) = e(D, g)</math></li> <li>7. Return 1 if all tests pass, 0 otherwise.</li> </ol> <p><b>KeyBitProofGen</b>(CRS, <math>\vec{r}, \theta, i</math>)</p> <ol style="list-style-type: none"> <li>1. Opens the <math>i</math>-th commitment of <math>\vec{C}</math>, that is <math>(\theta[i], r_i)</math> such that <math>C_i = g^{r_i} g^{\theta[i]s^i}</math>.</li> <li>2. Return <math>\text{open} \leftarrow (\theta[i], r_i)</math>.</li> </ol> <p><b>KeyBitCheck</b>(CRS, <math>\text{open}, i</math>)</p> <ol style="list-style-type: none"> <li>1. Parse <math>\text{open}</math> as <math>\text{open} = (b, r)</math></li> <li>2. Check that <math>C_i = g^r g_i^b</math>.</li> </ol> <p><b>EncSigDecrypt</b>(<math>\gamma, \theta</math>)</p> <ol style="list-style-type: none"> <li>1. Parse <math>\gamma</math> as <math>\gamma = (D, \vec{C}, \pi, \vec{PK}, \vec{\sigma})</math>.</li> <li>2. Parse <math>\vec{\sigma}</math> as <math>\vec{\sigma} = (\vec{\sigma}', r_\sigma)</math>.</li> <li>3. Compute <math>\sigma' = \vec{\sigma}'^{\frac{1}{\theta}}</math>.</li> <li>4. Return <math>\sigma = (\sigma', r_\sigma)</math>.</li> </ol>
---

Figure 4: Implementation of the Fair Exchange protocol.

## 6 Conclusion

In this work we introduced a practical protocol to exchange short signatures [3] fairly without relying on a TTP. It seems our technique 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. Thus it seems interesting to use our techniques in order to obtain a general framework to build practical fair protocols using bilinear maps.

## References

- [1] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *Proceedings of the 4th ACM conference on Computer and communications security - CCS '97*, pages 7–17, New York, New York, USA, April 1997. ACM Press.
- [2] Manuel Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1(2):175–193, May 1983.
- [3] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21(2):149–177, 2008.
- [4] Dan Boneh and Moni Naor. Timed Commitments. pages 236–254, August 2000.
- [5] Ernest Brickell, David Chaum, Jeroen van De Graaf, and Ivan Bjerre Damgård. Gradual and verifiable release of a secret. In Carl Pomerance, editor, *CRYPTO*, pages 156–166. Springer-Verlag, 1987.
- [6] Jan Camenisch and Markus Stadler. Proof Systems for General Statements about Discrete Logarithms, 1997.
- [7] R Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing - STOC '86*, pages 364–369, New York, New York, USA, November 1986. ACM Press.
- [8] Richard Cleve and Gilles Brassard. Controlled Gradual Disclosure Schemes for Random Bits and Their Applications. In Gilles Brassard, editor, *Advances in Cryptology CRYPTO 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 573–588. Springer New York, New York, NY, July 1990.
- [9] Ivan Bjerre Damgård. Practical and provably secure release of a secret and exchange of signatures. pages 200–217, January 1994.
- [10] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, June 1985.
- [11] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO*, January 1986.
- [12] Juan A Garay, Philip D MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource Fairness and Composability of Cryptographic Protocols. In *Proc 3rd Theory of Cryptography Conference TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428. Springer, 2006.

- [13] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281, April 1988.
- [14] S. Gordon, Jonathan Katz, and Henri Gilbert. Partial Fairness in Secure Two-Party Computation. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 157–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [15] Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [16] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect Non-interactive Zero Knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [17] Qiong Huang, Duncan S. Wong, and Willy Susilo. The Construction of Ambiguous Optimistic Fair Exchange from Designated Confirmer Signature without Random Oracles. In *Public Key Cryptography*, 2012.
- [18] Aniket Kate, Gregory Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [19] Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing - PODC '03*, pages 12–19, New York, New York, USA, July 2003. ACM Press.
- [20] Torben Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology CRYPTO '91*, pages 129–140. 1992.
- [21] Benny Pinkas and Eli Biham. Fair Secure Two-Party Computation. In Eli Biham, editor, *Advances in Cryptology EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, page 647. Springer Berlin Heidelberg, Berlin, Heidelberg, May 2003.

## A Proof of theorem 3

**Proof.** (Sketch)

As the underlying NIZK arguments are perfectly complete, we can see by inspection that so is the fair exchange protocol. Let  $\mathcal{A}$  be the adversary that breaks the fairness of our protocol. We build the following adversary  $\mathcal{B}$ .

We distinguish between two types of adversaries  $\mathcal{A}$ :

- *Type I*: the adversary  $\mathcal{A}$  does not lie.

- *Type II*: the adversary  $\mathcal{A}$  lies: that is he forges values for the NIZK arguments and/or the commitments.

Note that both adversaries may abort during the execution of the protocol.

The simulator  $\mathcal{B}$  will choose with probability 1/2 to bet that  $\mathcal{A}$  is of type I and with probability 1/2 that  $\mathcal{A}$  is of type II.

What happens in case of an adversary of type I, is that the only way for  $\mathcal{A}$  to win his by breaking the signature scheme (without even caring about the values sent by  $\mathcal{P}_B$ ). This means the simulator  $\mathcal{B}$  will try to build an adversary that breaks the security of the signature scheme. The case of adversary of type II is more straightforward: as  $\mathcal{A}$  lies, the simulator will be able to break the soundness of some of the arguments involved or the binding property of the commitment scheme.

For adversary  $\mathcal{A}$  of type I,  $\mathcal{B}$  does the following.  $\mathcal{B}$  generates himself the CRS, thus knowing the trapdoor  $s$ . When  $\mathcal{A}$  chooses the message  $m_B$  to be signed,  $\mathcal{B}$  will not compute the signature because he wants to break the signature scheme. More concretely  $\mathcal{B}$ :

1. Runs  $\text{BMGen}(1^\kappa)$  to obtain  $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ .
2. Sets  $s \xleftarrow{R} \mathbb{Z}_p$ .
3. Computes  $\text{CRS} \leftarrow \langle (p, \mathbb{G}, \mathbb{G}_T, e, g), (g, g^s, g^{s^2}, \dots, g^{s^\kappa}) \rangle = \langle (p, \mathbb{G}, \mathbb{G}_T, e, g), (g_0, g_1, \dots, g_\kappa) \rangle$ .
4. Sets  $\theta_B \xleftarrow{R} \mathbb{Z}_p$ .
5. Sets  $r_\sigma \xleftarrow{R} \mathbb{Z}_p$ .
6. Sets  $D = (g^{m_B} uv^r)^{\theta_B}$ .
7. Sets  $\tilde{\sigma} = (g^{\theta_B}, r_\sigma)$ .
8. Sets  $(r_1, r_2, \dots, r_\kappa) \xleftarrow{R} \mathbb{Z}_p^\kappa$ .
9.  $\vec{C} = (C_i)_{i \in [\kappa]} = (g^{r_i} g_i^{\theta^{[i]}})_{i \in [\kappa]}$ .

Note that all the proofs can be computed because  $\mathcal{B}$  knows  $s$ . Moreover  $\mathcal{A}$  is fooled because we have  $e(\tilde{\sigma}, g^{m_B} uv^r) = e((g^{m_B} uv^r)^{\theta_B}, g) = e(D, g)$ .

Some subtle issue arises though: Let us consider the case where  $\mathcal{A}$  detects that he is inside the simulator because he does a brute force attack on the remaining bits and realizes that  $\mathcal{B}$  does not know the signature. In this case  $\mathcal{A}$  will abort the simulation (not only the protocol) and in particular he will not to output the signature and thus our reduction will fail. As the commitment scheme is perfectly hiding and the arguments are perfectly zero-knowledge, we can deduce that this may only happen in round  $j$  where there are  $\kappa - j$  (few) remaining bits. Moreover  $\mathcal{A}$  runs a brute force attack, this means that  $\kappa - j \leq \log(Q(\kappa))$  where  $Q(\cdot)$  is a polynomial. Let  $\sigma_B$  be the signature that is output by  $\mathcal{B}$ . As  $\mathcal{A}$  does not lie we also have that  $\Pr[\text{SVf}(pk_A, m_A, \sigma_B) = \text{valid}] \geq \frac{1}{2^{\log(Q(\kappa))+1}}$  (+1 because  $\mathcal{A}$  has got one bit of advantage). We can deduce that the advantage of  $\mathcal{A}$  is such that  $\text{Adv}^{\text{FE}}(\mathcal{A}, \kappa, \lambda) \leq \frac{1}{2^{-(\log(Q(\kappa))+1)} 2^\kappa} = \frac{2^{\log(Q(\kappa))+1}}{2^\kappa} = \frac{2Q(\kappa)}{2^\kappa}$  which is still negligible, thus  $\mathcal{A}$  did not win, so our reduction is still valid. Consider now the case where  $\mathcal{A}$  does not abort in the simulation. That is he may abort the protocol but will finally output the signature  $\sigma_A$ , which an attempt to be a valid signature for message  $m_B$ . We can deduce that  $\text{Adv}^{\text{FE}}(\mathcal{A}, \kappa, \lambda) \geq \epsilon \Rightarrow$

$$\frac{\Pr[\text{SVf}(pk_B, m_B, \sigma_A) = \text{valid}]}{\Pr[\text{SVf}(pk_A, m_A, \sigma_B) = \text{valid}] 2^\kappa} \geq \epsilon \Rightarrow \Pr[\text{SVf}(pk_B, m_B, \sigma_A) = \text{valid}] \geq \epsilon$$

because we trivially have that  $\Pr[\text{SVf}(pk_A, m_A, \sigma_B)] \geq \frac{1}{2^\kappa}$ . Thus  $\mathcal{B}$  has broken the signature scheme.

For adversary  $\mathcal{A}$  of type II, the simulator  $\mathcal{B}$  does the following: he asks  $\mathcal{A}$  to compute  $\gamma_A$  which is parsed as  $(D_A, \vec{C}_A, \pi_{A,1}, \pi_{A,2}, \vec{P}\vec{K}_A, \vec{\sigma}_A)$ . For the proof of knowledge  $\mathcal{A}$  will use  $\mathsf{H}(\cdot)$  a random oracle which is controlled by  $\mathcal{B}$ . So  $\mathcal{B}$  will rewind  $\mathcal{A}$  and obtain  $\vec{P}\vec{K}'_A$  from which he will extract  $r_{A_i}$  and  $\theta_A[i]$  for each  $i \in [\kappa]$ . As  $\mathcal{A}$  lied,  $\mathcal{B}$  will be able to break the soundness of one of the NIZK argument or the binding property of the commitment scheme.

■

## B Proof of theorem 1

**Proof.**

**Completeness.** Let  $i \in [\lambda]$ . It's clear that the prover can compute  $A_i = C_i^{s^{\lambda-i}}$ . Then if  $b_i := B[i] \in \{0, 1\}$  we have  $b_i(b_i - 1) = 0$  and

$$\begin{aligned} e(A_i, C_i g_i^{-1}) &= e(g^{(r_i + b_i s^i) s^{\lambda-i}}, g^{r_i} g^{(b_i - 1) s^i}) \\ &= e(g^{(r_i s^{\lambda-i} + b_i s^\lambda)(r_i + (b_i - 1) s^i)}, g) \\ &= e(g^{r_i^2 s^{\lambda-i} + r_i b_i s^\lambda + r_i (b_i - 1) s^\lambda + b_i (b_i - 1) s^{\lambda+i}}, g) \\ &= e(g^{r_i^2 s^{\lambda-i} + s^\lambda r_i (2b_i - 1)}, g) \\ &= e(B_i, g) \end{aligned}$$

We can see that the prover can compute  $B_i$  as he knows  $r_i$  and  $b_i$  and the group elements  $g_{\lambda-i}, g_\lambda$  are public.

**Computational Soundness.** Assume there exist some adversary  $\mathcal{A}$  that breaks the soundness of the scheme (that is  $\mathcal{A}$  is able to open the commitment to some vector that does not contain bits) for at least one  $i$ . We build the following adversary  $\mathcal{B}$  that breaks the  $\lambda - polyDH$  assumption.  $\mathcal{B}$  receives the  $\lambda - polyDH$  tuple challenge  $(g, g_1, g_2, \dots, g_\lambda)$ .  $\mathcal{B}$  runs  $\mathcal{A}$  using the tuple as the CRS and obtains  $C_i, r_i, b_i$  such that  $C_i = g^{r_i} g_i^{b_i}$  and  $\pi_i = (A_i, B_i)$  where  $e(C_i, g_{\lambda-i}) = e(A_i, g)$  (1) and  $e(A_i, C_i g_i^{-1}) = e(B_i, g)$  (2).

From (1) we can deduce that  $A_i = C_i^{s^{\lambda-i}}$ . From (2) (as seen in the correctness proof) we have that  $B_i = g^{r_i^2 s^{\lambda-i} + r_i (2b_i - 1) s^\lambda + b_i (b_i - 1) s^{\lambda+i}}$

$\mathcal{B}$  can compute  $X = g^{r_i^2 s^{\lambda-i} + r_i (2b_i - 1) s^\lambda}$ , so it can obtain  $g_{\lambda+i} = (B_i \cdot X^{-1})^{\frac{1}{b_i(b_i-1)}}$  where  $b_i(b_i - 1) \neq 0$  as  $b_i \notin \{0, 1\}$ , and thus the  $\kappa - polyDH$  assumption is broken.

**Perfect zero-knowledge.** We observe first that the argument is perfectly witness-indistinguishable because for a given commitment  $C_i = g^{r_i} g_i^{b_i}$  there is only one possible argument that satisfies equations  $A_i = C_i^{s^{\lambda-i}}$  and  $e(A_i, C_i g_i^{-1}) = e(B_i, g)$ .

We describe now the zero-knowledge simulator. It generates the common reference string correctly, and also learns the trapdoor so it can create commitments that can be opened to any value. As the commitment can be opened and the trapdoor is known it is easy to compute an argument  $A_i, B_i$ .

Let us justify why the simulator simulates perfectly the real argument. Consider the *hybrid* stateful algorithm where the simulator generates the trapdoor and the common reference string but opens the commitment to real witness  $(r_i, b_i)$ . As the commitment is perfectly trapdoor the real argument is perfectly indistinguishable from the hybrid algorithm. Finally as the argument is perfect witness-indistinguishable the hybrid is perfectly indistinguishable from the simulated argument.

■



## C Proof of theorem 2

**Proof.**

**Perfect completeness.** The reason why the prover can compute  $U = (\frac{\prod_{i \in [\kappa]} C_i}{g^r})^{\frac{1}{s}}$  without knowing  $s$  is because  $U = \prod_{i \in [\kappa]} g^{\theta[i]s^{i-1}}$ . Indeed  $U$  corresponds to the vector  $(0, \theta[1], \dots, \theta[\kappa])$  that is moved by one position to the left. Similarly  $V$  can be computed because the prover knows the coefficients  $W[i]$  of the polynomial  $W(\cdot)$  of degree  $\kappa - 2$ , so we have  $V = \prod_{i \in [\kappa-1]} g_{i-1}^{W[i]}$ . The rest follows by simple inspection.

**Computational Soundness.** Let  $\mathcal{A}$  be the PPT adversary that breaks the soundness of the scheme. We build the following adversary  $\mathcal{B}$ .  $\mathcal{B}$  receives the challenge tuple  $(g, g_1, g_2, \dots, g_\kappa)$ . This challenge tuple stands for the CRS and is sent to  $\mathcal{A}$ .  $\mathcal{A}$  returns the following values:

- $\theta \xleftarrow{R} \mathbb{Z}_p$ .
- $\theta^*$  such that  $D = g^{\theta^*}$ .
- $(r_i, \theta[i]) \in (\mathbb{Z}_p \times \{0, 1\})^\kappa$  for  $i \in [\kappa]$  such that  $\vec{C} = (C_i)_{i \in [\kappa]}$  where  $C_i = g^{r_i} g^{\theta[i]s^i}$ .
- $\pi = (r, U, V) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}$ .

Assume first that  $r \neq \sum_{i \in [\kappa]} r_i \pmod p$  then we can deduce that  $U = (g^{\sum_{i \in [\kappa]} r_i - r} g^{\theta_i s^i})^{1/s}$ . As  $g^{\frac{\theta_i s^i}{s}} = g^{\theta_i s^{i-1}}$  is easily computable due to the fact that  $\theta[i]$  are known  $\mathcal{B}$  can deduce  $g^{\frac{\sum_{i \in [\kappa]} r_i - r}{s}}$  and as  $\delta = \sum_{i \in [\kappa]} r_i - r \neq 0 \pmod p$  is known,  $\mathcal{B}$  can compute  $g^{\frac{1}{s}} = (\frac{U}{g^{\delta/s}})^{\frac{1}{\delta}}$  and thus the  $\kappa$ -SDH assumption is broken. From now on we assume that  $\sum_{i \in [\kappa]} r_i = r$ . As the adversary  $\mathcal{A}$  wins, this means that there exists some  $j \in [\kappa]$  such that  $\theta[j] \neq \theta^*[j]$ . Moreover as the verification involving  $V$  passes we have that  $V = g^{\frac{\sum_{i \in [\kappa]} \theta[i]s^{i-1} - \theta^*[i]2^{i-1}}{s-2}}$ . As the decomposition in binary is unique we have that  $\Delta = \sum_{i \in [\kappa]} 2^{i-1}(\theta[i] - \theta^*[i]) \neq 0 \pmod p$ . We can rewrite  $V$  as  $V = g^{\frac{\Delta}{s-2} + \frac{\sum_{i \in [\kappa]} \theta[i]s^{i-1} - \theta^*[i]2^{i-1}}{s-2}}$  =  $g^{\frac{\Delta}{s-2} + \frac{\sum_{i \in [\kappa]} \theta[i](s^{i-1} - 2^{i-1})}{s-2}}$  =  $g^{\frac{\Delta}{s-2} + Z(s)}$  where the coefficients of  $Z(\cdot)$  are efficiently computable by  $\mathcal{B}$  because  $\forall i \in [\kappa] : s - 2 | s^{i-1} - 2^{i-1}$ . As  $\Delta \in \mathbb{Z}_p$  is also known this means  $\mathcal{A}$  can compute  $g^{\frac{1}{s-2}} = (\frac{V}{g^{Z(s)}})^{\frac{1}{\Delta}}$  and thus the  $\kappa$ -SDH assumption is broken.

**Perfect zero-knowledge.** The simulator works as follows. He generates the common reference string CRS correctly and saves the trapdoor  $s$ . Given the statement  $D, \vec{C} = (C_i)_{i \in [\kappa]}$  such that  $\vec{C}$  is formed by Pedersen commitments to bits in positions  $1, \dots, \kappa$  and such that  $D$  and  $\vec{C}$  are equivalent with respect to bases  $(2, s)$ , The simulator choose a random  $r' \in \mathbb{Z}_p$  and reveals it as the randomness of  $\prod_{i \in [\kappa]} C_i$ . Then the simulator sets  $D' = \frac{\prod_{i \in [\kappa]} C_i}{g^{r'}}$ ,  $U = (\frac{\prod_{i \in [\kappa]} C_i}{g^{r'}})^{\frac{1}{s}}$  and  $V = (\frac{U}{D})^{\frac{1}{s-2}}$ .  $r', U, V$  are indistinguishable from a real argument as they are all uniform. ■