Delegateable Signature Using Witness Indistinguishable and Witness Hiding Proofs

Chunming Tang[1,*] Dingyi Pei[1,2] Zhuojun Liu[3]
1 Institute of Information Security of Guangzhou University, P.R.China
2 State Key Laboratory of Information Security, Chinese Academy of Sciences, P.R.China
3 Academy of Mathematics and systems science, Chinese Academy of Sciences, P.R.China

Abstract

A delegateable signature scheme is a signature scheme where the owner of the signing key(Alice) can securely delegate to another party(Bob) the ability to sign on Alice's behalf on a restricted subset S of the message space. Barak first defined and constructed this signature scheme using non-interactive zero-knowledge proof of knowledge(NIZKPK)[1]. In his delegateable signature scheme, the function of NIZKPK is to prevent the signing verifier from tell which witness(i.e. restricted subset) is being used.

Witness indistinguishable(WI) and witness hiding(WH) proof systems are weaker proof model than zero-knowledge proof and were proposed by Feige and Shamir in [2], however, the verifier cannot also distinguish the witness which is being used in these two protocols. In this paper, we construct delegateable signature scheme using WI and WH proof protocols.

Keywords: Digital signature, delegateable signature, zero-knowledge proof system, witness indistinguishable proof, witness hiding proof

1 Introduction

A new Cryptographic tool, *a delegateable signature scheme*, is introduced by Barak[1]. Informally, a delegateable signature scheme is a signature scheme where the owner of the signing key(Alice) can securely delegate to another party(Bob) the ability to sign on Alice's behalf on a restricted subset S of the message space. By secure delegation we require both that this other party(Bob) would not be able to generate a signature for a message $m \notin S$ and that the signatures generated by the delegated party(Bob) would be indistinguishable from those generated by the owner of the signature key(Alice). As usual, we also require that any other party will not be able to generate a signature for any message m that it has not seen a signature for.

According to the motivation for delegateable signature, Alice can give Bob a *restricted sign key* in a delegateable signature. The restricted signing key $s_{Alice,T}$ will allow its holder to sign only messages that fit the template T. Formally, We'll say that a template T is a boolean predicate defined over the set of all strings (i.e. $T : \{0,1\}^* \to \{0,1\}$). We say that a message m fits the template T if $T(m) = 1$. In such a scheme we call Alice's key s_{Alice} the master signing key(as opposed to a restricted signing key). We require two properties from a delegateable signing scheme:

1. The holder of the master signing key can create a *T-restricted signing key* for any function $T : \{0,1\}^* \to \{0,1\}$. We say that s_T is a T-restricted signing key if using s_T one is able to sign any message m that satisfies $T(m) = 1$ and no other messages.

*Email: *chunming.tang@gmail.com* or *tangcm622@hotmail.com*

2. IF T, T' are two functions and m is a message such that $T(m) = T'(m) = 1$ then it is infeasible to distinguish between a signature that was generated using s_T and a signature that was generated using $s_{T'}$.

In [1], Barak provided the formal definition for delegateable signature schemess, and a construction using Non-interactive zero-knowledge proof of knowledge(NIZKPK)(which is based on trapdoor permutations) that meet this definition.

Witness indistinguishable(WI) and witness hiding(WH) of knowledge were introduced by Feige and Shamir[2]. Informally, a two party protocol in which party A uses one of several secret witnesses to an NP assertion is *witness indistinguishable* if party B cannot tell which witness A is actually using. The protocol is *witness hiding* if by the end of the protocol B cannot compute any new witness which he did not know before the protocol began. WH is a natural security requirement, and can replace zero-knowledge(ZK) in many cryptographic protocols.

Obviously, a protocol is WI and WH if it is ZK. Feige and Shamir had proven that the WI for NP relation R is a WH if R is a non-trivial(that is, R has at least two witness). Based on one way functions, they constructed the interactive witness indistinguishable(IWI) and interactive witness hiding(IWH) proofs, furthermore, they also provided the way for constructing noninteractive witness indistinguishable(NIWI) and noninteractive witness hiding(NIWH) proofs using noninteractive zero-knowledge proofs.

In delegateable signature scheme, if T, T' are two functions and m is a message such that $T(m) = T'(m) = 1$ then it is infeasible to distinguish between a signature that was generated using s_T and a signature that was generated using $s_{T'}$. In order to obtain the above requirement, NIZKPK is used so that the signing verifier cannot distinguishable which function(T or T') is being used[1]. In fact, it is feasible to obtain the above requirement using WI and WH proofs, this is because in the two protocols the verifier cannot also distinguish the witness which is being used.

> In Barak's delegateable signature, the zero-knowledge property of NIZKPK only guarantees the indistinguishability between function T and T', however, it is useless to other properties of delegateable signature scheme.

In this paper, we construct delegateable signature using NIWI and NIWH proofs. Barak's delegateable signature is based on trapdoor permutation, however, our signature is at most based on trapdoor permutation, and is probably based on weaker assumption than trapdoor permutation because NIWI and NIWH are weaker protocols than ZK(however, up to now, no one can construct NIWI and NIWH using weaker assumption than trapdoor permutation).

2 Notation and Definitions

By standard digital signatures we mean a signature scheme that is secure against adaptive existential forgery attack[3].

The standard notion of boolean circuits is used. We will identity boolean circuits with their canonical representation as strings in $\{0,1\}^*$. The size of a circuit C(denoted by $|C|$) is the size of this representation. If C is a circuit with n inputs then we denote by $L(C)$ the following language, $L(C) = \{x \in \{0,1\}^n | C(X) = 1\}$. For $m \in \{0,1\}^*$, we denote by C_m the canonical circuit of size $10|m|$ such that $L(C_m) = \{m\}$. We assume that any circuit C of n inputs is of size at least $10n$.

An algorithm can be either a uniform algorithm, modeled by a Turing machine, or a non-uniform algorithm, modeled by a sequence of circuits for each input length. Out adversaries in this paper will be polynomial non-uniform algorithms, which are modeled by a sequence of circuits $\{C_n\}_{n\in\mathbb{N}}$ such that $|C_n| \leq p(n)$ for some polynomial $p(\cdot)$.

An oracle algorithm is defined in the standard way(either as a Turing machine with a special oracle query tape, or as a sequence of circuits with oracle gates). If A is an oracle algorithm, and B is an algorithm then by $A^B(x)$ we denote the result of running A on input x with oracle access to B. If A is an algorithm that takes two inputs then by $A(x, \cdot)$ we denote the algorithm that is the result of fixing A's first input to be x. $V_P(x)$ denotes V's output after interaction with P on common input x(both of P and V are probabilistic polynomial time interactive Turing machine). $M(x, A)$(where A may be either P or V) denotes algorithm M's output on input x, where M may use algorithm A as a (blackbox) subroutine. Each call M makes to A is counted as one computation step for M.

If D is a distribution then $x \leftarrow D$ means that x is a random variable distributed according to D. If S is a set then $x \leftarrow_R S$ means that x is a random variable distributed according to the uniform distribution on the elements of S.

A negligible function is a function that grows slower that inverse of any polynomial. That is, $\mu : \mathbb{N} \to \mathbb{N}$ is negligible if for any positive polynomial $p(\cdot)$ there exists a number n_0 such that $\mu(n) < \frac{1}{p(n)}$ for all $n > n_0$. We will sometimes use $negl(\cdot)$ to denote some unspecified negligible function.

For any strings $x, y \in \{0,1\}^*$ we denote by $x \circ y$ the concatenation of x and y.

The value $\perp$ is a special symbol that denotes "failure".

2.1 Defining delegateable signature

We describe the definition for delegateable signature defined by Barak[1].

Assume a template $T : \{0,1\}^* \to \{0,1\}$ to be a boolean circuit C that computes T, so we talk about C-restricted keys instead of talking about T-restricted keys. Identifying circuits with their canonical representation as strings in $\{0,1\}^*$, therefore when saying something like "run the algorithm A with circuit C as input" it means that A is run on input the canonical representation of C as a string in $\{0,1\}^*$.

The delegateable signature scheme is usually composed of four algorithms:

1. The *key generator* algorithm(ab. $KEYGEN$). It generates the master signing and verifying keys. By the master signing key we mean the key that allows signing on any message(as well as creation of restricted signing keys). This is in contrast with a restricted signing key. The notation is $(s_{master}, v_{master}) \leftarrow KEYGEN(1^n)$.

2. The *delegation* algorithm(ab. $DELEGATE$). It is the algorithm creating a restricted signing key. This algorithm takes as input the master signing key s_{master} and a boolean circuit C, the output of $DELEGATE$ is a restricted signing key s_C that allows only to sign messages $m \in \{0,1\}^*$ such that $C(m) = 1$(or equivalently, $m \in L(C)$). The notation is $s_C \leftarrow DELEGATE(s_{master}, C)$.

3. The *signing* algorithm(ab. $SIGN$). It takes as input a message m and a restricted key s_C. If $C(m) = 1$ then the output of $SIGN$ would be a signature α. The notation is $\alpha \leftarrow SIGN(s_C, m)$.

4. The *verification* algorithm(ab. $VERIFY$). It takes as input the master verification key v_{master}, a message m, and an alleged signature α. It then outputs 1 if an only if α is a valid signature of m with respect to the key v_{master}.

The following definition presents the requirements from the four algorithms described above.

Definition 1 *A* **delegateable signature scheme** *is a quadruple of algorithm* $(KEYGEN, DELEGATE, SIGN, VERIFY)$ *that satisfies the following three requirements:*

1. *(Validity) For any circuit C and message $m \in \{0,1\}^*$ such that $m \in L(C)$, if we perform the following experiment:*

   (a) *Generate master keys: $(s_{master}, v_{master}) \leftarrow KEYGEN(1^n)$*

   (b) *Create a restricted signing key s_C that allows to messages in $L(C) : S_C \leftarrow DELEGATE(s_{master}, C)$*

   (c) *Sign the message m using the restricted key $s_C : \tau \leftarrow SIGN(s_C, m)$.*

   Then it will always (i.e., with probability 1) hold that $VERIFY(v_{master}, m, \tau) = 1$.

2. *(Security against existential forgery) Consider an adversary A that is given the master verification key and access to two oracles. The first oracle, when given a circuit C, provides A with a restricted signing key s_C(corresponding to the above verification key). The second oracle, when given a circuit C' and a message m, provides A with a signature on m using $s_{C'}$. Suppose that the restricted keys that A received from its second oracle are $s_{C_1}, ..., s_{C_k}$, and the signed messages that A received from its second oracle are $m_1, ..., m_j$. Let $\mathcal{M}$ be the set of messages that S has acquired signatures for, either explicitly(from the second oracle) or implicitly (from the first oracle). That is, $\mathcal{M} = L(C_1) \cup ... \cup L(C_k) \cup \{m_1, ..., m_j\}$. It will be required that it is infeasible for A to output a signed message (m, τ), such that $VERIFY(v_{master}, m, \tau) = 1$ and $m \notin \mathcal{M}$.*

   The formal requirement would be as follows: for any polynomial non-uniform oracle algorithm A we perform the following experiment:

   (a) *generate the keys $(s_{master}, v_{master}) \leftarrow KEYGEN(1^n)$.*

   (b) *Run A with oracle to the delegation algorithm w.r.t. the master signature keys s_{master}. That is run $A^{DELEGATE(s_{master}, \cdot)}$.*

   Let (m, τ) be the output of A, we say that A is successful if $VERIFY(v_{master}, m, \tau) = 1$ and $m \notin \mathcal{M}$. We require that any polynomial non-uniform oracle algorithm A has only negligible probability of succeeding in this experiment.

3. *(Indistinguishable Signatures) If $m \in \{0,1\}^*$ is a message and C_1, C_2 are two circuits of the same size such that $m \in L(C_1) \cap L(C_2)$ then we require that it would be infeasible to distinguish between a signature on m that was made using s_{C_1} and a signature on m that was made using s_{C_2}. The actual attack we are considering is even stronger: after having an access to the same oracle as in the previous item (the oracle that on input a circuit C outputs a restricted signing key s_C), the adversary A will output two circuits C_1, C_2 and a message m such that $C_1(m) = C_2(m) = 1$ and $|C_1| = |C_2|$. We then choose at random one of the two circuits C_b, create a restricted signing key s_{C_b} and use it to on the message m. We require that the probability that A manages to guess the which circuit was used is at most $\frac{1}{2}$. The formal requirement follows.*

   For any polynomial non-uniform oracle algorithm A we perform the following experiment:

   (a) *Generate the keys: $(s_{master}, v_{master}) \leftarrow KEYGEN(1^n)$*

   (b) *Run A with oracle to the delegating algorithm w.r.t. s_{master}(i.e. run A with oracle to $DELEGATE(s_{master,\cdot})$*

   (c) *At this stage A outputs is (m, C_1, C_2) where $C_1(m) = C_2(m) = 1$ and $|C_1| = |C_2|$*

   (d) *Choose $b \leftarrow_R \{1, 2\}$*

   (e) *Generate a restricted signing key s_{C_b}: Let $s_{C_b} \leftarrow DELEGATE(s_{master}, C_b)$*

(f) *Provide A with $SIGN(s_{C_b}, m)$*

(g) *Let $b' \in \{0,1\}$ be the final output of A*

A is successful in this experiment if $b = b'$. We requirement that A is successful with only negligible advantage over half. That is we require that the probability that after performing this experiment b' is equal to b is at most $\frac{1}{2} + negl(n)$.

Remark 1 *For boolean circuits and standard digital signatures, the reader may make reference to [1].*

2.2 Witness indistinguishable and witness hiding protocols

The concepts of witness indistinguishable and witness hiding were introduced by Feige and Shamir and constructed by them under the existence of one way functions[2]. Both notions seem weaker than zero-knowledge, yet they suffice for some specific applications.

We first introduce the following definitions, then introduce the definitions of WI and WH.

Definition 2 *Let R be a relation $\{(x, w)\}$ testable in polynomial time, where $|x| = |w|$. For any x, its witness set $w(x)$ is the set of w such that $(x, w) \in R$.*

Definition 3 *An interactive proof of knowledge system for relation R is a pair of algorithms (P, V) satisfying:*

1. *Completeness: $\forall (x, w) \in R,\ Prob(V_{P(x,w)}(x)$ accepts$) > 1 - \mu(n)$*
2. *Soundness: $\exists M \forall P' \forall x \forall w'\ Prob(V_{P'(x,w')}(x)$ accepts$) < Prob(M(x, w'; P') \in w(x)) + \mu(n)$*

The probability is taken over the coin tosses of V, P', and M. The knowledge extractor M runs in expected polynomial time, and uses P' as a blackbox.

Remark 2 *If $w(x)$ is empty, this definition implies that the probability that V accepts is negligible.*

Definition 4 *Proof system (P, V) is zero knowledge(ZK) over R if there exists a simulator M which runs in expected polynomial time, such that for any probabilistic polynomial time V', for any $(x, w) \in R$, and any auxiliary input y to V', the two ensembles $V'_{P(x,w)}(x, y)$ and $M(x, y; V')$ are polynomially indistinguishable. M is allowed to use V' as a subroutine.*

2.2.1 Witness indistinguishable proofs

Now, we introduce witness indistinguishability defined in [2]. Informally, a protocol is witness indistinguishable if the verifier cannot tell which witness the prover is using(even if the verifier knows all witnesses to the statement being proved).

Definition 5 *Proof system (P, V) is witness indistinguishable (WI) over R if for any V', any large enough input x, any $w_1, w_2 \in w(x)$, and for any auxiliary input y for V', the ensembles, $V'_{P(x,w_1)}(x, y)$ and $V'_{P(x,w_2)}(x, y)$, generated as V's view of the protocol, are indistinguishable.*

Unlike definition for Zero-knowledge, the definition for WI involves no simulator M.

The following propositions hold for WI.

Proposition 1 *WI is preserved under polynomial composition of protocols, however, ZK is not done.*

Remark 3 *If (P, V) is any ZK protocol, then the protocol is WI.*

Proposition 2 *Under the assumption that one-way functions exist, any NP language has a constant round WI proof system.*

2.2.2 Witness hiding proofs

We then introduce witness hiding protocol defined in [2]. The concept of Witness Hiding(WH-to be defined shortly) is possible alternative to zero-knowledge, it is a weaker requirement than zero-knowledge, but in many cases, it still satisfies the security demands of cryptographic protocols. Informally, a protocol (P, V) is WH if participating in the protocol does not help V to compute any new witnesses to the input which he did not know at the beginning of the protocol. This is a natural security requirement of cryptographic protocols. In order to prove the WH property, one must show that if V' can compute a witness to the input after participating in the interactive proof, then he had this capability in him even before the protocol began. The definition of WH involves a probability distribute over the inputs.

Definition 6 *G is a generator for relation R if on input 1^n it produces instances $(x, w) \in R$ of length. G is an invulnerable generator if any polynomial time nonuniform cracking algorithm C, $Prob((x, C(x)) \in R) < \nu(n)$, where $x = G(1^n)$. The probability is taken over the coin tosses of G and C.*

Definition 7 *Let (P, V) be a proof of knowledge system for relation R, and let G be a generator for this relation. (P, V) is WH on (R, G) if there exists a witness extractor M which runs in expected polynomial time, such that for any nonuniform polynomial time V'*

$$Prob(V'_{P(x,w)}(x) \in w(x)) < Prob(M(x; V', G) \in w(x)) + \nu(n)$$

where $x = G(1^n)$. The probability is taken over the distribution of the inputs and witness, as well as the random tosses of P and M. The witness extractor is allowed to use V' and G as blackboxes.

Proposition 3 *Let G be a generator for relation R. Then under the assumption that one way functions exist, there exists a witness hiding proofs.*

There are two main differences between WH and zero-knowledge: 1) The distribution on the inputs enters the definition (through G). There might be infinitely many inputs on which P willingly discloses his witness, but the protocol may still be WH if the probability of G picking an inputs is negligible. This distribution on the inputs implies that V' must have the same auxiliary input for any common input of size n, unlike the case of ZK protocols, where V's auxiliary input may depend upon x. 2) The definition only guarantees that "whole" witness are not disclosed. Partial information may leak. In particular, the communicatoin tape generated by $V'_{P(x,w)}(x)$ may not be simulatable in random polynomial time, and thus may serve as evidence that the protocol took place. In some cases this is an advantage. For example: Digital signatures cannot be zero-knowledge(otherwise they are forgeable) and thus zero-knowledge is an adequate framework for defining their security. On the other hand, digital signatures can be witness hiding, hiding the auxiliary information which allows the true signer to sign messages.

2.2.3 Constructions of witness indistinguishable and witness hiding proofs

Witness indistinguishable proof systems are not necessary witness hiding. For example, any language with unique witness has a proof system that yields the unique witness(and so may fail to be witness-hiding), yet this proof system is trivially witness independent. On the other hand, for some relations, witness indistinguishability implies witness hiding, provided that the prover is probabilistic polynomial-time.

Proposition 4 *Let G be a generator for a proper claw free function f, which generates pairs (x, w) where $x = f(w)$, with uniform distribution over the arguments w. Let (P, V) be a proof of knowledge system for proving knowledge of a pre-image of x. Then if (P, V) is WI over over f, then it is WH over (f, G).*

Claw free functions are rare. Many candidates for intractable functions do not even have two premages. Feige and Shamir had showed a transformation which transforms any relation R to a new relation R^2 for which each argument has two independent witness.

Given relation $R = \{(x, w)\}$, define R^2, where $((x_1, x_2), w) \in R^2$ iff $(x_1, w) \in R$ or $(x_2, w) \in R$. Given a generator G for R, obtain a generator G^2 for R^2, by applying G twice independently, and discarding at random one of the two witness.

Proposition 5 *Let G be a (or an invulnerable) generator for relation. Let (P, V) be a proof of knowledge system for R^2(P proves knowledge of a witness of one of two instances in R). Then if (P, V) is WI over R^2, then it is WH over (R^2, G^2).*

In order to construct WI proofs of knowledge which are also WH, we composed two random instances of the NP language. This gives a new NP language, and thus has zero-knowledge protocols. These protocols are also witness indistinguishable(). WI is preserved even if the basic steps are composed in parallel(). By (), these parallel protocols are also witness hiding on G^2.

Proposition 6 *Let G be a generator for relation R. Then under the assumption that one way functions exist, R^2 has a constant round proof of knowledge which is witness hiding over (R^2, G^2).*

2.2.4 Non-interactive witness indistinguishable and witness hiding proofs

Obviously, any NIZK proofs is non-interactive witness indistinguishable and witness hiding.

Definition 8 *Noninteractive proof system (P, V) is witness indistinguishable over R if for any large enough input x, any $w_1, w_2 \in w(x)$, and for a randomly chosen public string σ, the ensembles $P(x, w_1, \sigma)$ and $P(x, w_2, \sigma)$, generated as P's proof indistinguishable. The probability space is that of the random choices of σ together with P's random coin tosses.*

Proposition 7 *Let (P, V) be a noninteractive proof system which is witness indistinguishable over R. Then the system remains witness indistinguishable even if polynomially many proofs are given using the same public random string σ.*

Proposition 8 *Let G be a generator for relation R. Let (P, V) be a noninteractive proof system for R^2 (P proves knowledge of a witness of one of two instances in R). Then if (P, V) is WI over R^2, then it is WH over (R^2, G^2).*

As a result, there exists noninteractive witness indistinguishable and witness hiding proofs.

3 Constructing delegateable signature using WI and WH proofs

In this section, we will construct two delegateable signature schemes. A standard signature scheme, a Witness indistinguishable protocol, and a witness hiding protocol are used as main tools in constructing these schemes.

Before constructing delegateable signature, we recall the main idea behind the construction, which was described by Barak in [1].

3.1 The main idea

Consider the following attempt at constructing a delegateable signature signature using a standard (non-delegateable) signature scheme $(KEYGEN', SIGN', VERIFY')$:

1. The master signing and verifying keys (s_{master}, v_{master}) would be the signing and verifying keys (s, v) of the standard signature scheme $(KEYGEN', SIGN', VERIFY')$.

2. To delegate the ability to sign on message in $L(C)$, the holder of the master signing key will sign on the circuit C. That is, the restricted signing key s_C will be the couple $< C, SIGN'(s, C) >$.

3. The signing algorithm, when given a restricted key $s_C =< C, \tau >$ and a message m such that $C(m) = 1$, would simple output s_C.

4. The verifying algorithm, when given a message m and an alleged signature $< C, \tau >$ on m, would check that $C(m) = 1$ and $VERIFY'(v, C, \tau) = 1$.

Barak remarked that this scheme satisfy property 2(security against existential forgery) of Definition 1, however, it obviously does not satisfy property 3(indistingushable signatures) of that. This is because a signature using a restricted key s_C contains the description of the circuit C itself, and so clearly if $|C| = |C'|, m \in L(C) \cap L(C')$ but C is not equal to C' then one can distinguish between a signature on m using s_C and a signature on m using $s_{C'}$. In [**?**], Barak solved this problem by the following way: when signing on a message m, the signing algorithm, instead of outputting $< C, \tau >$ as the signature, proves in NIZKPK that it knows a couple $< C, \tau >$ such that $C(m) = 1$ and $VERIFY'(v, C, \tau)$.

In WI and WH proofs, it is well known that the verifier can not distinguish that which of two witness is beng used by the prover, hence, we can use WI and WH proofs(non-interactive) to replace the NIZKPK proofs in Barak's delegateable signatures.

We must stress that all NP relation R in next section has at least two witnesses, that is, R is non-trivial.

3.2 Delegatelable signature using WI proofs

Recall that a delegateable signature scheme is composed of four algorithms ($KEYGEN$, $DELEGATE$, $SIGN$, $VERIFY$). In our construction we use a standard(non-delegatelable) signature scheme $(KEYGEN', SIGN', VERIFY')$ and a WI proofs for some non-trivial NP relation R.

Construction 1 *A delegateable signature scheme using WI proofs.*

1. *To generate the master signing and verification keys, we'll use the standard signature scheme's key generator to obtain a pair of signing and verification keys (s, v). We'll also use the generator to generator a reference string Σ. The master signing key s_{master} will be the signing key s. The master verification key v_{master} will be the concatenation of the verification key v and the reference string Σ. Actually, for technical reasons, we add the master verification key to the master signing key. Formally, this means that the algorithm $KEYGEN$ is defined as follows:*

$$(s_{master}, v_{master}) = (s \circ v \circ \Sigma, v \circ \Sigma) \leftarrow KEYGEN(1^n).$$

2. *To delegate the ability to sign on a set $L(C)$ where C is some circuit, the holder of the master signing key signs on the circuit C. That is the restricted signing key s_C is the concatenation of C and a signature (using s) on C. We'll also add to s_C the public*

information $v_{master} = v \circ \Sigma$. *Formally, the algorithm DELEGATE is defined in the following way:*

$$DELEGATE(s_{master}, C) = DELEGATE(s \circ v \circ \Sigma, C) \rightarrow C \circ \tau \circ v \circ \Sigma = s_C,$$

where $\tau \leftarrow SIGN'(s, C)$.

3. *If a party has some restricted signing keys* s_C*, and wishes to sign on a message* m *such that* $C(m) = 1$*, then this party proves using a* **witness indistinguishable proof** *that it knows a circuit* C *such that* $C(m) = 1$ *and a signature on* C*. That is, the signature on a message* s *is a witness indistinguishable proof for relation* R' :

$$(< m, v >, < C, \tau >) \in R' \ iff \ C(m) = 1 \wedge VERIFY'(v, C, \tau) = 1.$$

Like to Barak's delegateable signature, we modify R' *as* R *in order to make the witness' size bounded by some fixed polynomial in the size of the theorem, one needs to add the size of the circuit* C *to the theorem. We define the modified relation* R *as following:*

$$(< m, v, 1^l >, < C, \tau >) \in R \ iff \ C(m) = 1 \wedge VERIFY'(v, C, \tau) = 1 \wedge |C| = l.$$

We use a NIWI for this relation R*. Given that, the formal definition of the algorithm* $SIGN$ *is the following:*

$$SIGN(C \circ \tau \circ \Sigma, m) \leftarrow PROVE_{NIWI}(\Sigma, < m, v, 1^{|C|} >, < C, \tau >) \circ 1^{|C|}$$

4. *To verify the validity of a signature* $\alpha = \pi \circ 1^l$ *on a message* m *with respect to the master verification key* $v_{master} = v \circ \Sigma$*, one simply verifies that* π *is a valid WI proofs, that is, the verifier believes that the signer holds a circuit* C*(a witness) satisfying* $C(m) = 1$ *and* $VERFY'(v, C, \tau) = 1$*. We describe this verification process by following way:*

$$VERIFY(v \circ \Sigma, m, \pi \circ 1^l) = 1 \ iff \ VERPROOF(\Sigma, < m, v, 1^l >, \pi) = 1$$

Theorem 1 *The tuple* $(KEYGEN, DELEGATE, SIGN, VERIFY)$ *of Construction 1 is a delegateable signature scheme.*

Proof: Construction 1 is a delegateable signature scheme if it satisfies the three properties required in Definition 1.
(i) Proof of the validity: From the definition of the relation R and character of the NIWI proof, the validity of construction 1 holds.
(ii) Proof of security against existential forgery:[1] Let A be an oracle algorithm. Suppose we run with A the experiment described in Property 2 of Definition 1, which is hereafter called Experiment 1. Suppose that the output of A is (m, α). To prove security against existential forgery we meed to prove the following claim:

> **Claim 1.1:** Let $s_{C_1}, ..., s_{C_k}$ be the restricted signature keys that A received from its oracle. Recall that A is successful if $VERIFY(v_{master}, m, \alpha) = 1$ and $m \notin \mathcal{M}$ where $\mathcal{M} = L(C_1) \cup ... \cup L(C_k)$. The probability that A is successful is negligible.
> *Proof:* Suppose, toward the contradiction, that A is successful, with non-negligible probability ϵ. We'll build an adversary A' that manages to break the standard signature scheme $(KEYGEN', SIGN', VERIFY')$ with probability at least $\frac{\epsilon}{2}$. This will contradict the security of the standard signature scheme(and so we'll be done).

[1] we copy the proof in [1], because it can completely prove the security against existential forgery of our delegateable signature.

Recall that to break the standard signature scheme, algorithm A' gets as input the verification key v and oracle access to the signing algorithm $SIGN(s, \cdot)$. The goal of A' would be the output a pair of a message and a signature (m, τ) such that $VERIFY'(m, \tau) = 1$ but m was not queried by A' from its oracle. We now describe the algorithm A':

Algorithm A':

1. Input: Verification key v(where $(s, v) \leftarrow KEYGEN'(1^n)$)
2. Oracle access: oracle $S(\cdot)$ where $S(m) = SIGN'(s, m)$

A's operation:

1. Let $(\Sigma, aux_S, aux_E) \leftarrow GEN(1^n)$.
2. We let $v_{master} = v \circ \Sigma$, $s_{master} = s \circ v \circ \Sigma$. (Note that A' only knows v_{master} and does not know all of s_{master}.
3. Algorithm A' run algorithm A with input v_{master}. Note that the distribution of v_{master} is identical to the distribution of A's input in Experiment 1.
4. Suppose that the i^{th} query A makes to its oracle is the circuit C_i. To answer it, A' first uses the oracle S to produce $\tau_i \leftarrow S(C_i) = SIGN(s, C_i)$. Algorithm A' then replies to A with $S_{C_i} = C_i \circ \tau_i \circ v \circ \Sigma$. Note that this is again exactly the same answer that A would get in the Experiment 1, when is has an oracle to $DELEGATE(s_{master}, \cdot)$.
5. Let (m, α) be the final output of algorithm A, where $\alpha = \pi \circ 1^l$. (Note that if A is successful then we know that $VERIFY(v_{master}, m, \alpha) = 1$ which means that $VERPROOF(\Sigma, < m, v, 1^l >, \pi) = 1$.
6. Using the exacter EXT, algorithm A' obtains $< C, \tau > \leftarrow EXT(\Sigma, < m, v, 1^l >, \pi, aux_E)$ and outputs (C, τ).

As both the input A gets and the answers it receives from its oracle are distributed exactly the same as in Experiment 1 we know that with probability ϵ, algorithm A will be successful. This means that A will output a pair (m, α) such that $m \notin \mathcal{M}$ and $VERIFY(v_{master}, m, \alpha) = 1$. Recall that if $C_1, ..., C_k$ are the queries that A made, then $M = L(C_1) \cup ... \cup L(C_k)$. Since $VERPROOF_{NIWI}(\Sigma, < m, v, 1^l >, \pi) = 1$, by the proof of knowledge property of the NIWI, the output of the extractor $< C, \tau >$ satisfies in the case $(< m, v, 1^l >, < C, \tau >) \in R$ with probability at least $1 - \frac{\epsilon}{2}$. In the case, $C(m) = 1$ and $VERIFY'(, v, C, \tau) = 1$. We claim that it cannot be the case that $C = C_i$ for some i. This is because if this was the case then we would have that $C_i(m) = 1$ and so $m \in L(C_i) \subseteq \mathcal{M}$. Therefore we see that with overall probability $\frac{\epsilon}{2}$ we have actually succeeded in breaking the standard signature scheme $(GEN', SIGN', VERIFY')$.

Note that in order to prove this claim we did not need to use indistinguishability property of the NIWI, but rather only its proof of knowledge property. ∎

(iii) Proof of indistinguishable signature: Let A be an oracle algorithm. Recall the experiment described in property 3(Indistinguishable signature) of Definition 1(We copy it here for the reader's convenience):

Experiment 2:

1. Generate the keys: $(s_{master}, v_{master}) \leftarrow KEYGEN(1^n)$

2. Run A with oracle to the delegating algorithm w.r.t. s_{master}(i.e. run A with oracle to $DELEGATE(s_{master}, \cdot)$).

3. At this stage A outputs is (m, C_1, C_2) where $|C_1| = |C_2|$ and $C_1(m) = C_2(m) = 1$.

4. Choose $b \leftarrow_R \{1, 2\}$.

5. Generate s restricted signing key s_{C_b}: Let $s_{C_b} \leftarrow DELEGATE(s_{master}, C_b)$.

6. Provide A with $SIGN(s_{C_b}, m)$.

7. Let $b' \in \{0, 1\}$ be the final output of A.

To prove this property we need to prove the following claim:

Claim 2 *The probability that $b' = b$ is at most $\frac{1}{2} + negl(n)$.*

Proof: Assume that the probability that $b' = b$ is larger than $\frac{1}{2} + negl(n)$, obviously, It is a contradiction against indistinguishability of NIWI. ∎

Remark 4 *The proof of security against existential forgery is equal to the proof of security against existential forgery in delegateable signature constructed by Barak using NIZKPK. This is because only the extractor of knowledge is needed, however, the zero-knowledge property of NIZKPK and indistinguishability property of NIWI is not needed.*

3.3 Delegateable signature using WH proofs

Let a delegateable signature scheme be composed of four algorithms ($KEYGEN$, $DELEGATE$, $SIGN$, $VERIFY$), and ($KEYGEN'$, $SIGN'$, $VERIFY'$) be a standard(non-delegatelable) signature scheme.

Notation For any strings $x, y \in \{0, 1\}^*$ we denote by $x \circ y$ the concatenation of x and y.

Construction 2 *A delegateable signature scheme using WH proofs.*

1. *To generate the master signing and verification keys, we'll use the standard signature scheme's key generator to obtain a pair of signing and verification keys (s, v). We'll also use the generator to generator a reference string Σ. The master signing key s_{master} will be the signing key s. The master verification key v_{master} will be the concatenation of the verification key v and the reference string Σ. Actually, for technical reasons, we add the master verification key to the master signing key. Formally, this means that the algorithm $KEYGEN$ is defined as follows:*

$$(s_{master}, v_{master}) = (s \circ v \circ \Sigma, v \circ \Sigma) \leftarrow KEYGEN(1^n).$$

2. *To delegate the ability to sign on a set $L(C)$ where C is some circuit, the holder of the master signing key signs on the circuit C. That is the restricted signing key s_C is the concatenation of C and a signature (using s) on C. We'll also add to s_C the public information $v_{master} = v \circ \Sigma$. Formally, the algorithm $DELEGATE$ is defined in the following way:*

$$DELEGATE(s_{master}, C) = DELEGATE(s \circ v \circ \Sigma, C) \rightarrow C \circ \tau \circ v \circ \Sigma = s_C,$$

where $\tau \leftarrow SIGN'(s, C)$.

3. *If a party has some restricted signing keys s_C, and wishes to sign on a message m such that $C(m) = 1$, then this party proves using* **a witness hiding proof** *that it knows a circuit C such that $C(m) = 1$ and a signature on C.*

   In order to use NIWH proofs, we must construct a new NP relation R^2. We random select two circuit C_1 and C_2 which both of them satisfy $C_i(m) = 1$ and $VERIFY'(v, C_i, \tau_i) = 1$, then construct NP relation R^2 :

   $$(< m, v, 1^l >, < C_1, C_2, \tau >) \in R^2)\ iff$$

   $$\exists i\ s.t.\ (C_i(m) = 1 \wedge VERIFY'(v, C_i, \tau_i) = 1 \wedge |C_i| = l).$$

   Where $i = 1, 2$.

   We use a NIWH for this relation R^2. Given that, the formal definition of the algorithm $SIGN$ is the following:

   $$SIGN(C_1 \circ C_2 \circ \tau \circ \Sigma, m) \leftarrow PROVE_{NIWH}(\Sigma, < m, v, 1^l >, < C_1, C_2, \tau >) \circ 1^l$$

4. *To verify the validity of a signature $\alpha = \pi \circ 1^l$ on a message m with respect to the master verification key $v_{master} = v \circ \Sigma$, one simply verifies that π is a valid WH proofs, that is, the verifier believes that the signer holds a circuit C(a witness) satisfying $C(m) = 1$ and $VERFY'(v, C, \tau) = 1$. We describe this verification process by following way:*

   $$VERIFY(v \circ \Sigma, m, \pi \circ 1^l) = 1\ iff\ VERPROOF(\Sigma, < m, v, 1^l >, \pi) = 1$$

Remark 5 *It is feasible in step 3 of Construction 2 if only one of C_1 and C_2 satisfies $C(m) = 1 \wedge VERIFY'(v, C, \tau) = 1$. This is the reason that the prover can also convince the verifier that $(< m, v, 1^l >, < C_1, C_2, \tau >) \in R^2$ if he only knows a witness C_1 or C_2. In fact, NP relation $(R')^2$ can be denoted by*

$$(< m, v, 1^l >, < C_1, C_2, \tau >) \in R^2)\ iff$$

$$(C_1(m) = 1 \wedge VERIFY'(v, C_1, \tau_1) = 1 \wedge |C_1| = l) \text{ or } (C_2(m) = 1 \wedge VERIFY'(v, C_2, \tau_1) = 1 \wedge |C_2| = l).$$

Theorem 3 *The tuple $(KEYGEN, DELEGATE, SIGN, VERIFY)$ of Construction 2 is a delegateable signature scheme.*

Proof: The Proof of the validity and security against existential forgery equal to those of Construction 1. This is because the witness hiding is not necessary in these proofs.
Proof of indistinguishable signature: We still use Experiment 2.
Experiment 2:

1. Generate the keys: $(s_{master}, v_{master}) \leftarrow KEYGEN(1^n)$
2. Run A with oracle to the delegating algorithm w.r.t. s_{master}(i.e. run A with oracle to $DELEGATE(s_{master}, \cdot)$).
3. At this stage A outputs is (m, C_1, C_2) where $|C_1| = |C_2|$ and $C_1(m) = C_2(m) = 1$.
4. Choose $b \leftarrow_R \{1, 2\}$.
5. Generate s restricted signing key s_{C_b}: Let $s_{C_b} \leftarrow DELEGATE(s_{master}, C_b)$.
6. Provide A with $SIGN(s_{C_b}, m)$.
7. Let $b' \in \{0, 1\}$ be the final output of A.

To prove this property we need to prove the following claim:

Claim 4 *The probability that $b' = b$ is at most $\frac{1}{2} + negl(n)$.*

Proof: It is well known that a NIWH proof, which is constructed in Proposition 5 or 8, is a NIWI proof. Now, assume that the probability that $b' = b$ is larger than $\frac{1}{2} + negl(n)$, obviously, It is a contradiction against indistinguishability of NIWI. ∎

4 Extensions and open problems

In [1], Barak extended his construction in several ways, such as, (i) relaxing the dependence of the signature's size and the restricted key's size on the size circuit; (ii) allowing delegation of sets in **NP**/Poly; (iii) discouraging sharing of the restricted key; (iv) making signatures indistinguishable even for the delegating user; (v) adding more levels of delegation. In fact, our constructions can also be extended in similar ways. Furthermore, our delegateable signature schemes can also construct "anonymous unlinkable certificates".

We construct delegateable signatures using WI and WH protocols, however, some problems about WI and WH are fuzzy, and are presented as open problem by us.

1. As usual, a WH proof is not WI, then, in what conditions it is WI?
2. How is a WI proof constructed directly no using ZK proofs? or how is a WH proof constructed no using ZK proofs or WI proofs?
3. if there exists weaker assumption(such as one way function) for constructing NIWI and NIWH proofs than trapdoor permutation.

In this paper, a WH protocol is a WI protocol because the WH protocol in Proposition 5 and 8 are constructed by WI protocol, however, the first open problem tell us that the common WH protocol is probably not WI protocol. So, our delegateable signature must use the WH protocol which is constructed in Proposition 5 or 8.

If there exists weaker assumption(such as one way function) for constructing NIWI and NIWH proofs than trapdoor permutation, we will obtain the delegateable signature basing on weaker assumption than trapdoor permutation.

5 Conclusion

In this paper, we present the way constructing delegateable signature schemes using NIWI and NIWH proofs. However, up to now, the NIWI and NIWH protocols can not be constructed based on weaker assumption than trapdoor permutation, so our signature scheme are also based on trapdoor permutation.

References

[1] B. Barak, *Delegateable Signatures*, 2001. http://www.math.ias.edu/ boaz/Papers/delgsigs.ps

[2] U. Feige, A. Shamir, *Witness Indistinguishable and Witness Hiding Protocols*, In 22nd ACM Symposium on the Theory of Computing, 416-426, 1990

[3] S. Goldwasser, S. Micali, and R. Rivest. *A digital signature scheme secure against adaptive chosen-message attacks.* SIAM Journal of Computing, vol. 17, no. 2, April 1988, pp. 281–308