Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent

(EXTENDED ABSTRACT)

Shafi Goldwasser* Rafail Ostrovsky†

TR-92-083

December 1992

Abstract

The standard definition of digital signatures allows a document to have many valid signatures. In this paper, we consider a subclass of digital signatures, called *invariant* signatures, in which all legal signatures of a document must be identical according to some polynomial-time computable function (of a signature) which is hard to predict given an unsigned document. We formalize this notion and show its equivalence to non-interactive zero-knowledge proofs.

*MIT. This research was supported in part by NSF-FAW CCR-9023313, NSF-PYI CCR-865727, Darpa N0014-89-J-1988, BSF 89-00312.

†International Computer Science Institute at Berkeley and University of California at Berkeley. Supported by NSF Postdoctoral Fellowship. Parts of this work were done at MIT, Bellcore and IBM T.J. Watson Research Center.

1 Introduction

Currently, due to the lack of proven non-trivial lower bounds on NP problems, the theory of cryptography is primarily based on unproven assumptions such as the difficulty of particular computational problems such as integer factorization, or more generally the existence of one-way and trapdoor functions. It is thus naturally desirable to establish minimal complexity assumptions for basic cryptographic primitives, and to establish connections among these primitives. Indeed, it has been an active and in many cases successful area of research. For example, pseudo-random generators [BM] were shown to be equivalent to the existence of any one-way function [ILL, H]. On the other hand, several other primitives, such as secret-key exchange seem to require the trapdoor [IR] property.

Digital signatures have been an especially interesting case in point. Originally introduced by Diffie and Hellman [DH], the first implementation was based on the RSA trapdoor function [RSA] which yields a deterministic signature scheme where each document has a unique valid signature. Later, the notion of digital signatures which are secure against chosen message attack[1] was formally defined by [GoMiRi] and proved to exist under a sequence of decreasingly weaker assumptions: the existence of claw-free permutations [GoMiRi] (e.g. factoring), the existence of trapdoor permutations [BeMi], the existence of one-way permutations by [NY], and finally the existence of one-way functions by [Ro]. In all of these schemes, each document may have many valid signatures.

The fact that digital signatures can be implemented if one-way functions exist without the need for a trapdoor [NY, Ro] is somewhat remarkable, as by definition a digital signature seems to posses the essential flavor of a trapdoor function: namely, it should be easy for everyone to verify the correctness of a signature, while it should be hard for everyone except a privileged user (with access to the private file) to sign. In this paper, we study which aspects of digital signatures allows for this dichotomy and whether digital signatures can in some cases be used in cryptographic protocols instead of trapdoor functions.

We show that the issue of having many different valid signatures of the same document plays a role in the above question. That is, on the positive side, we show that digital signatures can sometimes be used instead of trapdoor functions, provided that all valid signatures of the same document have an *invariant* property which is unpredictable from the document itself. On the negative side, we show that this *invariant* property for a signature scheme may require a trapdoor for its implementation (unless non-interactive zero-

[1] Note that RSA does not satisfy security against adaptive chosen message attack as there do exist messages for which the signature can be forged.

knowledge proofs among polynomial-time participants can also be implemented without a trapdoor).

Invariant signatures are interesting in their own right, as they capture the flavor of having a unique valid signature per document as in the case of RSA, and yet can be proven secure against adaptive chosen message attack as in the case of [GoMiRi, BeMi, NY, Ro]. Achieving these two aspects simultaneously may prove valuable in applications.

1.1 Invariant Signatures

Let us recall the definition of digital signatures as defined in [GoMiRi]. Informally, the setting is as follows: in a network, every user can generate (using a polynomial-time algorithm) a pair of keys: the public key and the corresponding secret key. In addition to the generation algorithm, the signature scheme is provided with two probabilistic polynomial-time algorithms: one for signing and one for verifying. Given an arbitrary document, a user applies his signing algorithm to the document, his public key, and his secret key. Given a signature of a document, any other user can verify the validity of the signature by applying the polynomial time verification algorithm to the signature, document, and the public key of the signer. No adversary can forge a signature for a new document, even after asking for arbitrary signature samples in an adaptive fashion.

The additional constraint we put on digital signatures so as to make them *invariant*, is (informally) that there exists a deterministic poly-time computable function g computed on signatures such that with high probability (1) for any document D and for any two legitimate signatures $\sigma_1(D)$ and $\sigma_2(D)$, $g(\sigma_1(D)) = g(\sigma_2(D))$ and (2) given D, $g(\sigma(D))$ is pseudo-random. If the above conditions hold we say that the signature scheme is invariant under g.

Although not the subject of this paper, we suggest that our definition of invariant signatures might serve as a good definition for what we may want from a *finger print* of a document: hard to predict for any document even in an adaptive setting, dependent perhaps on the time of inquiry, and yet unique.

1.2 Non-Interactive Zero-Knowledge Proofs and Digital Signatures

We investigate the comparative difficulty of non-interactive zero-knowledge proofs ($\mathcal{NIZK}$) [BFM] and digital signatures ($\mathcal{DS}$) [GoMiRi]. These seemingly different primitives were shown to be connected in a paper by Bellare and Goldwasser [BG], where it was shown that the existence of one-way functions and non-interactive zero-knowledge proofs implies the existence of digital signatures (secure against adaptive chosen-message attacks). We remark that the known constructions of non-interactive zero-knowledge proofs with polynomial-time

participants use the *trapdoor* permutations assumption [FLS], while digital signatures can be implemented based on *any* one-way function [Ro].

We show that the existence of invariant digital signatures is equivalent to the existence of non-interactive zero-knowledge proofs. That is, we show that while a signature scheme in which a document can be signed in an unconstrained plurality of ways requires the existence of any one-way function, a signature scheme in which each document has unique or at least "similar signatures" (according to any "nontrivial" poly-time computable function — this is the invariant property!) requires the same assumptions as non-interactive zero-knowledge proofs (i.e. currently the trapdoor assumption is necessary).

More precisely, we consider non-interactive zero-knowledge proofs in the random string model, where users in the system can read a pre-existing common (polynomial size) random string set up by the system (a model defined by [BFM]). We prove that in this common random string model, the existence of invariant digital signatures is equivalent to the existence of non-interactive zero-knowledge proofs for any hard to predict NP language (see definition in 2.2). To prove this theorem we must define invariant signatures in the common random string model.

1.3 A simple example: using digital signatures to achieve asymmetry

Suppose two probabilistic polynomial-time players (Alice and Bob) wish to agree on a boolean predicate $B(\cdot)$, so that when later given a randomly chosen x as a common input, Bob can *not* predict $B(x)$ with probability (over x and Bob's coin tosses) bounded away from half, but Alice can compute $B(x)$ and convince Bob of the value of $B(x)$. Under what assumptions can we implement such a protocol?

Before we examine the above question, let us recall definitions of a one-way function and a trapdoor function. Informally, a poly-time computable function f is one-way if when we pick x uniformly at random and compute $y \leftarrow f(x)$, it is infeasible for any polynomial time machine to find x' in $f^{-1}(y)$ for a non-negligible fraction of the instances. Again informally, a trapdoor function, is a one-way function with an additional secret key, the knowledge of which makes inversion easy.

Assuming the existence of one-way trapdoor permutations, Alice and Bob can achieve the above task. In particular, they can agree on a trapdoor one-way permutation (f, f^{-1}), so that Alice knows (f, f^{-1}) and Bob knows only f. In addition, they agree on a hard-core [GL] bit $B(\cdot)$ for f. (Notice that Alice and Bob must make sure that f is really a permutation for $B(\cdot)$ to be well defined.) Subsequently, when x is given, Alice can invert f and compute a hard-core bit, while Bob can not.

Can we achieve the above task using one-way functions which are not trapdoor? Let us

examine if digital signatures (which do not need trapdoor in their implementation) might be useful.

At first glance, to implement a simple protocol specified above could be done using digital signatures as follows: Alice prepares a public and a secret key (of a signature scheme), gives her public key to Bob and convinces him that her public key is produced using an appropriate key-generation algorithm. Moreover, they agree on a hard-core bit B of a signature for any document x'. Notice that given x and a public key of Alice, the signature of x is hard to find for any polynomial-time player, and thus Bob can not predict the hard-core bit of a signature of x, while Alice can easily compute it. Since we can implement signatures based on one-way functions (without the trapdoor) it seems that we can implement the above protocol without the trapdoor... What is wrong in this argument?

The problem, is that this bit is *not* well defined. That is, the specification of digital signatures allows for many legal signatures of x. However, if we put an additional constraint on the digital signature scheme, then the above argument will go through. The additional constraint is to have an *invariant* signature scheme (as above). Then, to implement the above game, Alice can use a hard-core bit of $g(\sigma(D))$ (where all signatures of D are invariant under g) and the bit is well-defined. Thus, notice that *invariant* digital signatures can be used in the above setting instead of a trapdoor function.

2 Model and Definitions

2.1 Negligible, noticeable and infeasible functions

We use the usual O, o and $1/o(1)$ (asymptotically tending to ∞) notation. We fix some function $s(n) = n^{1/o(1)}$ and call it *infeasible*. We call $\epsilon(n) = 1/s^{O(1)}(n)$ *negligible* and $\delta(n) = 1/O(n^c), c > 0$ *noticeable*. In this case, n is a security parameter, which we omit when clear from the context. We use standard definitions of one-way functions and computationally indistinguishable distributions (see, for example,[GL, ILL, H]). If S is a probability space then $x \leftarrow S$ denotes the algorithm which assigns to x an element randomly selected according to S. For probability spaces $S, T, \ldots$, the notation $\Pr(p(x, y, \cdots) : x \leftarrow S; y \leftarrow T; \cdots)$ denotes the probability that the predicate $p(x, y, \cdots)$ is true after the (ordered) execution of the algorithms $x \leftarrow S$, $y \leftarrow T$, etc. The notation $\{f(x, y, \cdots) : x \leftarrow S; y \leftarrow T; \cdots\}$ denotes the probability space which to the string σ assigns the probability $\Pr(\sigma = f(x, y, \cdots) : x \leftarrow S; y \leftarrow T; \cdots)$, f being some function. If S is a finite set we will identify it with the probability space which assigns to each element of S the uniform probability $\frac{1}{|S|}$. (Then $x \leftarrow S$ denotes the operation of selecting an element of S uniformly at random).

2.2 Non-Interactive Zero-Knowledge ($\mathcal{NIZK}$) Proofs in the Common Random String Model

Non-interactive zero-knowledge proofs were introduced in [BFM]. We note that this is where the "common random string model" was introduced as well.

Common random string model: at the time of the system set-up a string of a fixed (polynomial in the security parameter) length is chosen uniformly at random and published by a trusted center for everyone in the system (provers, verifiers, users etc.) such that it can be read but not modified.

Informally, a $\mathcal{NIZK}$ proof of an $\mathcal{NP}$ statement in a common random string model is a way for any polynomial-time user to convince other users that some statement is true without revealing anything else. That is, given a common random string, and a witness to an $\mathcal{NP}$ statement, there should be a probabilistic poly-time algorithm (for the prover) which constructs a proof of that statement, and a probabilistic poly-time algorithm (for the verifiers) to check that the proof is correct. Moreover, such proof should not reveal anything about the witness.

Formally, the following definition is essentially taken from [BDMP].

Definition 1 We fix an $\mathcal{NP}$ language L (with poly-time relation $\rho(\cdot,\cdot)$ and constant d such that $x \in L$ iff $\exists w, |w| < |x|^d, \rho(x,w) = 1$.) We say that two probabilistic polynomial-time algorithms $(prover(\cdot,\cdot,\cdot), verifier(\cdot,\cdot,\cdot))$ constitute **bounded** $\mathcal{NIZK}$ for language L if the following conditions are satisfied: there exist a polynomial l such that

Completeness: For all $x \in L$, $|x| = n$, sufficiently large n, and ϵ negligible, where w is such that $|w| < n^d$ and $\rho(x,w) = 1$, the $\Pr(verifier(x,w,c) = accept : c \leftarrow \{0,1\}^{l(n)}; y \leftarrow prover(x,w,c))$
$> 1 - \epsilon(n)$.

(Here, c is the "common random string", w is the NP witness, and y is the output of the prover which is computed non-interactively. The probability is taken over the choice of c and the prover's coin tosses).

Soundness: For all probabilistic polynomial-time players $prover'$, $x \notin L$, $|x| = n$, for sufficiently large n, and negligible ϵ, the
$\Pr(verifier(x,w,c) = accept : c \leftarrow \{0,1\}^{l(n)}; y \leftarrow prover'(x,c))$
$< \epsilon(n)$.

(Here, the probability is taken over the choice of c and prover's coin tosses).

Zero-Knowledge: There exists a probabilistic expected polynomial-time algorithm $S(\cdot,\cdot)$ such that for all $x \in L$, $|x| = n$, and w such that $|w| < n^d$ and $\rho(x,w) = 1$, for all probabilistic polynomial time algorithms D, for all sufficiently large n, the

$$|\Pr(D(c,x,y) = 1 : c \leftarrow \{0,1\}^{l(n)}; y \leftarrow prover(x,w,c)) -$$

$$\Pr(D(c,x,S(x,c)) - 1 : c \leftarrow \{0,1\}^{l(n)})| < \epsilon(n)$$

In the above c is called the "common random string", and l the length of the common random string.

REMARKS:

- One difference from above definition to [BDMP] is that we impose the soundness condition only on probabilistic polynomial-time $prover'$s. This is not actually necessary as known constructions achieve soundness against all $prover'$s. However, as in the context of this paper we show equivalence to a digital signatures in which a reasonable forger to consider is probabilistic polynomial time, we relax the soundness requirement here as well.

- The above definition is specified for a single theorem of a fixed polynomial size. This **bounded** $\mathcal{NIZK}$ definition can be extended to polynomially-many theorems each of polynomial length and to many users in the roles of both provers and verifier. This is the notion of $\mathcal{NIZK}$ we adopt here. To modify the above definition to accommodat ethis extension, we must require (as in [BDMP]) the existence of many pairs of $prover_i, verifier_i$ for which completeness and soundness are true, and change the zero-knowledge condition as follows.

  [Zero-Knowledge$'$:] There exists a probabilistic expected polynomial time algorithm S such that for all $x_1, x_2, \ldots \in L \cap \{0,1\}^n$, where $|w_1|, |w_2|, \ldots < n^d$ and $\rho(x_1,w_1) = 1$, $\rho(x_2,w_2) = 1, \ldots$, for all probabilistic polynomial time algorithm D, for all sufficiently large n, for all negligible ϵ,
  $|\Pr(D(c,(x_1,y_1),(x_2,y_2),\ldots) = 1 : c \leftarrow \{0,1\}^{l(n)}; y_1 \leftarrow prover_1(x_1,w_1,c);$
  $y_2 \leftarrow prover_2(x_2,w_2,c); \ldots) \quad -$
  $\Pr(D(c,(x_1,S(x_1,c)),(x_2,S(x_2,c)),\ldots) = 1 : c \leftarrow \{0,1\}^{l(n)})| < \epsilon(n)$.

- Another aspect of $\mathcal{NIZK}$ is a preservance of zero-knowledge in an adaptive setting, which means that even after requesting polynomially-many proofs one by one, the probability for polynomial-time Adv (over its coin-flips) of being able to distinguish

an NIZK proof of a new theorem from the run of the simulator is negligible. Notice that if $\mathcal{NIZK}$ proofs remains Zero-Knowledge even in an adaptive setting, then the statements may be *dependent* on the previous proofs and on the common random string. From now on, when we refer to $\mathcal{NIZK}$, we refer to $\mathcal{NIZK}$ which is secure in an adaptive setting. To modify the above definition to accommodate this extension we further refine the zero knowledge condition as follows.

[Zero-Knowledge$''$:] There exists a probabilistic expected polynomial time algorithm S such that for all polynomial time Adv, for all probabilistic polynomial time D, for all sufficiently large n, for all negligible ϵ,
$|\Pr(D(c,(x_1,y_1),(x_2,y_2),\ldots) = 1 : c \leftarrow \{0,1\}^{l(n)}; x_1 \leftarrow Adv(c);$
$y_1 \leftarrow prover_1(x_1,w_1,c); x_2 \leftarrow Adv(c,x_1,y_1); y_2 \leftarrow prover_2(x_2,w_2,c);\ldots) -$
$\Pr(D(c,(x_1,S(x_1,c)),(x_2,S(x_2,c)),\ldots) = 1 : c \leftarrow \{0,1\}^{l(n)}; x_1 \leftarrow Adv(c);$
$y_1 \leftarrow S(x_1,c); x_2 \leftarrow Adv(c,x_1,S(x_1,c)); y_2 \leftarrow S(x_2,c);\ldots)| < \epsilon(n).$

- We note that in our setting, provers are polynomial-time machines.

- An additional property of $\mathcal{NIZK}$ that we must stress is of being *publically verifiable* $\mathcal{NIZK}$ proof system, which means that the proof can be verified by any polynomial-time machine which has access to a common random string.

In [BFM, DMP1, BDMP] it was shown how $\mathcal{NIZK}$ could be implemented, based on algebraic assumptions. In [DMP2, KMO] the $\mathcal{NIZK}$ was implemented based on the general complexity assumptions and without a common random string, but at a price of a small pre-processing stage, which was interactive. Finally, in [FLS] it was shown how $\mathcal{NIZK}$ could be implemented without pre-processing, based on (verifiable) trapdoor one-way permutations. (In [BY], they show how verifiability requirement could be implemented based on trapdoor one-way permutations). Moreover, in [FLS] it was shown how to convert $\mathcal{NIZK}$ into publically-verifiable and adaptively secure (see remarks above) $\mathcal{NIZK}$ proof system. Again, we mention that it is not known how the assumptions (of one-way trapdoor permutations) could be reduced further.

Definition 2 *We say that a language L is* hard to predict *if there exist a probabilistic polynomial time algorithm $S(1^n)$ (which samples $X \in \{0,1\}^n$) such that for every probabilistic polynomial-time algorithm Adv, for all sufficiently large n and for all negligible ϵ, the probability (over S and Adv coin tosses) that Adv can correctly decide if $X \in L$ is less then $\frac{1}{2} + \epsilon(n)$.*

REMARK: The above definition can be modified as follows: we say that a language L is *sometimes hard to predict* if there exist a probabilistic polynomial time algorithm $S(1^n)$ (which samples $X \in \{0,1\}^n$) such that on a noticeable fraction H of $S(1^n)$, for every probabilistic polynomial-time algorithm Adv, for all sufficiently large n and for all negligible ϵ, the probability (over S and Adv coin tosses) that Adv on X in H can correctly decide if $X \in L$ is less then $\frac{1}{2} + \epsilon(n)$.

Definition 3 *We say that nontrivial $\mathcal{NIZK}$ exists, if there exists a (sometimes) hard to predict $L \in \mathcal{NP}$ which possesses an $\mathcal{NIZK}$ proof system.*

We note that the existence of $\mathcal{NIZK}$ proofs for (sometimes) hard to predict L implies the existence of uniform one-way functions [OW].

2.3 Invariant Digital Signatures ($\mathcal{INV} - \mathcal{DS}$)

The formulation of the digital signatures of [GoMiRi] allows any document to have many valid signatures (i.e. accepted by the signature verification algorithm as valid) of the same document. For invariant signatures we make the additional requirement that all valid signatures of the same document be "similar", that is, there exists an easy to compute function defined on signatures which yields the same value for all signatures of the same document. This function should be hard to compute from the document itself with access to the public key (but without access to the secret key).

In the following definition we incorporate the possibility that a common random string c was published by a trusted center at the time of a system set up for everyone in the system (signers and verifiers) to read but not to modify. This is similar to the set up of NIZK (see previous section). The definition of an invariant digital signature scheme can be made in the standard model as well (without the presumption of the existence of c), but as in this paper we show the equivalence of invariant signatures and NIZK in the common random string model, we present the definition of invariant digital signatures in this model. The polynomial $l(n)$ will denote the length of the common random string with security parameter n.

Definition 4 An **invariant signature scheme** π is a quadruple (G, S, V, g) such that the following conditions hold: let l be a polynomial function

G: is a probabilistic poly-time computable algorithm (the "key generation" algorithm) which on input 1^n (the security parameter), $c \in \{0,1\}^{l(n)}$ (the common random string) outputs a pair of strings (*secret-key, public-key*). We let the random variables $G_1(1^n)$ denote the first output and $G_2(1^n)$ the second output.

(Wlog we let $|G_1(1^n)| = |G_2(1^n)| = n$. The probability is over $c \leftarrow \{0,1\}^{l(n)}$ and G's coin tosses.)

S: is a probabilistic poly-time computable algorithm (the "signing" algorithm) which on input strings 1^n, $c \in \{0,1\}^{l(n)}$ (the common random string), $D \in \{0,1\}^*$ of length polynomial in n (the document), and a pair of strings $\{secret - key, public - key\}$ in the range of $G(1^n, c)$ outputs a string. The output is referred to as the "signature" of D (with respect to $public - key$ and c). When the context is clear we let $\sigma(D)$ denote an output of $S(1^n, D, G(1^n, c), c)$.

V: is a probabilistic poly-time computable algorithm (the "verification" algorithm) which receives as inputs the strings 1^n (the security parameter), $D \in \{0,1\}^*$ of length polynomial in n (the document), s (the presumed signature of D), $c \in \{0,1\}^{l(n)}$ and $public - key \in G_2(1^n)$, and outputs either true or false. We require that for all D in n, the $\Pr(V(1^n, D, s, public - key, c) = true : c \leftarrow \{0,1\}^{l(n)};$
$\{secret - key, public - key\} \leftarrow G(1^n, c); s \leftarrow S(1^n, D, \{secret - key, public - key\}, c)) = 1$

(Namely, signatures produced by the signing algorithm S are always accepted by the verifying algorithm V for any pair of public and private keys produced by key generation algorithm G).
If $V(1^n, D, s, public\text{-}key, c) = true$ then we say that s is a "valid" signature of D (with respect to $public - key$ and c).

security: Let F be a probabilistic poly time forging algorithm which receives as input the strings 1^n, $c \in \{0,1\}^{l(n)}$, and $public - key \in G_2(1^n)$; can request and receive signatures with respect to $public - key$ and c of polynomially-many adaptively chosen documents $\{D_i\}$ and finally outputs a pair of strings (D, s). Then, for all such F, for all sufficiently large n, for all negligible functions ϵ, the probability that F outputs (D, s) where $D \notin \{D_i\}$, and s is a valid signature of D with respect to $public - key$ and c is less than $\epsilon(n)$.

(The probability is taken over the outcome of G, signatures of D_i, and the coin tosses of F).

invariant function $g(\cdot, \cdot)$: is a polynomial time computable function which takes as input strings 1^n and s (when clear we use notation $g(s)$ for $g(1^n, s)$) and produces as output a string $t \in \{0,1\}^{r(n)}$ where r is a fixed polynomial, such that:

invariance Let Adv be a probabilistic polynomial-time algorithm which receives as input strings 1^n, $c \in \{0,1\}^{l(n)}$, and produces as output the tuple $(public -$

$key, D, \sigma_1(D), \sigma_2(D))$ where $public{-}key \in \{0,1\}^n$, and $\sigma_1(D)$ and $\sigma_2(D)$ are both valid signatures of D with respect to $public - key$ and c. Then, for all such Adv, for all $public - key \in \{0,1\}^n$, for any negligible ϵ, and for sufficiently large n, the probability that $g(\sigma_1(D)) \neq g(\sigma_2(D))$ is less than $\epsilon(n)$.

(Here the probability is taken over $c \leftarrow \{0,1\}^{l(n)}$, and the coin tosses of Adv.) (Note, that the definition implies that even the honest signer who has access to the secret key can not produce two signatures of the same document for which g is not the same with non-negligible probability.)

pseudo-randomness Let Adv be a probabilistic polynomial time algorithm which operates in two stages on input strings 1^n, $c \in \{0,1\}^{l(n)}$, and $public - key \in G_2(1^n)$. In the first stage Adv can request and receive signatures with respect to $public - key$ and c of polynomially-many (in n) adaptively chosen documents $\{D_i\}$. At the end of the first stage, Adv outputs a polynomial length string D not in $\{D_i\}$. In the second stage, Adv is presented with a string t on which it outputs 0 or 1 (we let $Adv(t)$ denote the output bit). Let $\alpha = \Pr(Adv(t) = 1 : c \leftarrow \{0,1\}^{l(n)}; \{secret - key, public - key\} \leftarrow G(1^n, c); s \leftarrow S(1^n, D, \{secret - key, public - key\}, c); t \leftarrow g(s))$ and let $\beta = \Pr(Adv(t) = 1 : t \leftarrow \{0,1\}^{r(n)})$. Then, for all Adv, for all negligible ϵ, for all sufficiently large n, $|\alpha - \beta| < \epsilon(n)$.

We call g the *invariant function* of the signature scheme, and l the length of the invariant function.

REMARK: We note that in the above definition the invariant property holds for *any* public file $public - key$, and not just over $G_2(1^n)$. This requirement ensures that *invariant* property holds for any public key, even a maliciously chosen one, and avoids problem pointed out in [BY] of lack of certification in [FLS].

The most important aspect of invariant signature scheme for our application is:

Lemma: If $\pi = (G, S, V, g)$ is an invariant signature scheme, then there exists a polynomial time computable Boolean predicate P which on input 1^n and s, outputs 0 or 1 such that the following conditions hold:

1. "P is invariant for all signatures of a document ": Let Adv be a probabilistic polynomial-time algorithm which takes as input strings $c \in \{0,1\}^{l(n)}$, and produces as output $(public{-}key, D, \sigma_1(D), \sigma_2(D))$ where $public - key \in \{0,1\}^n$, $\sigma_1(D), \sigma_2(D)$ are valid signatures of D with respect to $public - key$ and c. Them for all Adv, for all $public - key$, for all

negligible ϵ, for all sufficiently large n, the probability that $P(\sigma_1(D)) \neq P(\sigma_2(D)$ is less than $\epsilon(n)$. (The probability is taken over $c \leftarrow \{0,1\}^{l(n)}$ and coin-tosses of Adv.)

2. "P is unpredictable from D": Let Adv be a probabilistic polynomial time algorithm which receives as input strings $1^n, c \in \{0,1\}^{l(n)}, public-key \in G_2(1^n)$; can request and receive signatures with respect to $public-key$ and c of polynomially-many (in n) adaptively chosen documents $\{D_i\}$; and finally outputs a polynomially length string D not in $\{D_i\}$ and a bit b. Let $\alpha = \Pr(b = P(t) : c \leftarrow \{0,1\}^{l(n)}; \{secret-key, public-key\} \leftarrow G(1^n, c); s \leftarrow S(1^n, D, \{secret-key, public-key\}, c); t \leftarrow g(s))$. Then, for every Adv, for every negligible ϵ, and for all sufficiently large n, $|\alpha - \frac{1}{2}| \leq \epsilon(n)$.

We refer to the predicate P as, the *invariant property* of π.

This lemma follows immediately from the definition of invariant signature scheme.

REMARK: We must stress that digital signatures of [GoMiRi, BeMi, NY, Ro] are *not* known to be invariant in the above sense. In fact, while honest signer can sign in some predetermined (in fact, deterministic [G]) way, there exists many valid signatures for the same document which bear no similarity to each other. In contrast, invariant signatures require all valid signatures of a document to be "similar" according to some polynomial time computable function which is unpredictable from the document itself.

3 Preliminaries

Before we show the equivalence between the existence of $\mathcal{NIZK}$ and $\mathcal{INV}-\mathcal{DS}$, we review necessary ingredients of [FLS] and [BG] scheme.

3.1 Where Feige-Lapidot-Shamir use Trapdoor?

The [FLS] solution for $\mathcal{NIZK}$ for $\mathcal{NP}$ when the participants are polynomial-time requires the assumption that trapdoor permutations exist. This assumption is not necessary throughout their construction. In fact, the only place where the trapdoor property is used is to construct a "hidden random string". In particular, they show how to use a common random string in order to get a "hidden random string" as follows:

- prover picks a trapdoor one-way permutation (f, f^{-1}) and sends to the verifier the code of f. In addition, let B be a hard-core predicate associated with f [GL].
- A common random tape can be interpreted as a sequence of $(y_1, y_2, \ldots, y_m)$, with each $|y_i|$ of length n (a security parameter of f). Then *hidden random string* is defined as:

$(B(f^{-1}(y_1)), B(f^{-1}(y_2)), \ldots, B(f^{-1}(y_m)))$, where $B(\cdot)$ is a hard-core bit [GL]. Notice that since f is a permutation, the hidden random string is well-defined[2]. Notice that since f is a *trapdoor* permutation, the polynomial time prover can compute $f^{-1}(y_i)$.

Using different f's the prover can construct new hidden random bits (for each new theorem). Thus, they show how assuming a common fixed (polynomial length) random string and the existence of a trapdoor one-way permutations, a $\mathcal{NIZK}$ which is publically verifiable and Zero-Knowledge (in an adaptive setting) can be constructed for $\mathcal{NP}$.

3.2 Bellare-Goldwasser Signature Scheme

In [BG], it is shown how assuming publically verifiable non-interactive zero-knowledge proofs and pseudo-random functions of [GGM], a signature scheme can be constructed. (As was shown by [GGM], pseudo-random functions can be based on any one-way function.)

We outline their scheme below:

Step1: The signer chooses at random a seed s for a pseudo random function $F_s(\cdot)$ and publishes an encryption $E(s)$ along with the public information necessary to verify $\mathcal{NIZK}$ proofs (i.e., the random string etc.) as his public key, and keeps s as his secret key.

Step2: The signature of a document D is the value $v = F_s(D)$ together with an $\mathcal{NIZK}$ proof that indeed v was computed correctly.

We remark that in their construction the public-key contains the random string which is necessary for the signer for producing $\mathcal{NIZK}$ proofs. Since the signer serves here in the role of the prover, and it is to his advantage to chose the random string truly with uniform probability (else the chance of a successful forgery increases) the random string is made part of the signers public key rather than part of the systems choice.

In what follows, we will use a similar scheme except that the random string needed by the $\mathcal{NIZK}$ proof system will be specified by the system as a common random string.

4 The Equivalence of NIZK and INV-DS

Recall that when we say that nontrivial $\mathcal{NIZK}$ exist, we mean that $\mathcal{NIZK}$ proof system exist for some hard to predict language L. First, we state our main result:

MAIN THEOREM: $\mathcal{INV}-\mathcal{DS}$ exist if and only if nontrivial $\mathcal{NIZK}$ exist.

[2] In [FLS] it is assumed that f is a *verifiable* permutation. That is, verifier can check that it is a permutation by inspecting the code of f. In [BY], this is extended to arbitrary trapdoor one-way permutations.

Proof outline: We prove our main result in two parts: (1) $\mathcal{INV}-\mathcal{DS}$ imply the existence of nontrivial $\mathcal{NIZK}$; (2) nontrivial $\mathcal{NIZK}$ imply the existence of $\mathcal{INV}-\mathcal{DS}$;

First, we prove (1). We claim that digital signatures (and, hence, $\mathcal{INV}-\mathcal{DS}$) already imply the existence of a one-way function [Ro]. Thus, it remains to show that based on $\mathcal{INV}-\mathcal{DS}$ and the existence of one-way functions we can construct $\mathcal{NIZK}$ for some hard language. Assuming that one-way function f exist, we can construct a hard language in a straight-forward fashion. For example, let $L_f \triangleq \{x|B(f^{-1}(x))=1\}$, where B is hard core bit for f [GL]). We now give intuition for the fact that $\mathcal{INV}-\mathcal{DS}$ imply $\mathcal{NIZK}$ in the common random string model.

Let us first consider the case of one theorem $\mathcal{NIZK}$, with the common random string $R=(y_1,\ldots,y_m)$. To specify a hidden random string $H=(b_1,\ldots,b_m)$, instead of using a trapdoor function (i.e, $b_i = B\left(f^{-1}(y_i)\right)$ where B is a hard core bit as in section 3.1) the intuition is to use digital signatures (i.e., $b_i = P(\sigma(y_i))$ where P is some boolean function of the digital signature of y_i). Clearly, this intuition is correct if indeed for every y_i there exists a unique fixed boolean value b_i computable from any valid signature of y_i. Unfortunately, this is not the case for digital signatures in general [GoMiRi, BeMi, NY, Ro]. We remark that if it were true, then we could have implemented $\mathcal{NIZK}$ based on *any* one-way function instead of one-way trapdoor permutations as currently known. However, the above intuition is true for *invariant* digital signatures with high probability. That is, for *invariant* signatures it is the case that for all y_i there exist some invariant function g defined over the signatures $\sigma(y_i)$, and therefore an invariant Boolean predicate P defined over the signatures $\sigma(y_i)$.

Now, let us consider the case for many theorems. In this case, we need different hidden random strings for each new theorem. Thus, how do we extend the above intuition to obtain many hidden random strings for different theorems? (Recall that the solution of [FLS] was to pick a *new* trapdoor 1-way permutation f for each new proof so that a common random string $(y_1,y_2,\ldots,y_m)$, defines a hidden random string $(B(f^{-1}(y_1)),B(f^{-1}(y_2)),\ldots,B(f^{-1}(y_m)))$. The solution here is simple: when proving the i'th theorem T_i we use as a common hidden random string the sequence: $(P(\sigma(y_1+i)),P(\sigma(y_2+i)),\ldots,P(\sigma(y_m+i)))$. By adding a new i when proving each new theorem, we note that each new hidden random string is unpredictable even when given proofs of all the previous theorems. This is so, since the definition of $\mathcal{INV}-\mathcal{DS}$ requires that the hard bit $P(\sigma(y+i))$ be unpredictable in the adaptive setting (i.e., even if for all $j<i$, $P(\sigma(y+j))$ is given.)

We are now ready to outline how to use an $\mathcal{INV}-\mathcal{DS}$ to construct a $\mathcal{NIZK}$ for an $\mathcal{NP}$ language. Let n be a security parameter and nm is a length of a common random string (where m is as specified in [FLS]). (1) Run key-generation algorithm for $\mathcal{INV}-\mathcal{DS}$

m times and publish all m public keys as a "common" public key; (2) Keep a counter i (initialized to 0) of the number of theorems proven so far. (3) to prove theorem T_i utilize $(P(\sigma(y_1+i)), P(\sigma(y_2+i)), \ldots, P(\sigma(y_m+i)))$ as a hidden random sequence of the [FLS] construction.

Note that the completeness follows from that fact that both P and σ are efficiently computable, and the rest of the protocol is analogous to [FLS]. The soundness holds since the signature scheme we are using is *invariant*, and hence any particular choice of i with high probability specifies uniquely a hidden random string. Thus, for a sufficiently long random string, even if prover picks an arbitrary (but polynomially-bounded) i the conditions that at least one "block" has a property required by [FLS] proof do hold with very high probability (over common random string chosen with uniform distribution). The Zero-Knowledge property holds due to the fact that if the adversary can distinguish the $\mathcal{NIZK}$ and the simulator then [FLS] show that such a distinguisher can be turned into a good predictor of a hidden random bit. (The idea there is to use witness-indistinguishable proof that either the graph is Hamiltonian or that the first $2n$ random bits of the common random string a pseudo-random and are produced from a seed of length n. Exploring properties of witness-indistinguishability [FLS] show that the distinguisher of the simulator can be turned into a distinguisher for a pseudo-random generator or into a predictor of a hidden random bits.) In our construction, predicting a hidden random bit provides us with predictor of the *invariant property*, which by definition enables us to to forge a $\mathcal{INV}-\mathcal{DS}$ for some new D'. Since our signature scheme is secure against existential adaptive chosen-message attacks, we get a contradiction.

In order to show (2), we first note that $\mathcal{NIZK}$ for hard to predict L imply the existence of one-way functions [OW]. Hence, we must show that assuming one-way functions and $\mathcal{NIZK}$ proofs is sufficient to construct $\mathcal{INV}-\mathcal{DS}$. This, however, is essentially established for us by [BG] with the following modification of their construction. The idea is to make sure that $E(s)$ (of [BG] Step 1) uniquely specifies s, i.e., is a commitment to s. If this is the case, then for any document D, $F_s(D)$ (of [BG] Step 2) is uniquely defined, and the invariant function will be simply $F_s(D)$. Any bit of $F_s(D)$ can be used as a hard-core bit for the invariant predicate P (as discussed in the section on the definition of invariant signatures).

Now, we specify how to perform step 1 of [BG], based on any one-way function. In order to commit to a seed s, consisting of bits $s_1, s_2, \ldots, s_n$, the player commits *to each bit* s_i separately using a modification of Naor's scheme [N]. (The scheme of [N] is interactive, in which the player who receives committed bits (called Bob) chooses a random string during the conversation). In our protocol, the challenges of Bob are substituted by a (dedicated for this purpose) portion of the common random string. Following through an

argument analogous to [N] shows that this scheme uniquely determines s with overwhelming probability (over uniformly distributed common random string), and hence we can use the proof of security presented in [BG] here as well. Hence we are done with (2). $\square$

References

[BM] Blum M., and S. Micali "How to Generate Cryptographically Strong Sequences Of Pseudo-Random Bits" *SIAM J. on Computing,* Vol 13, 1984, pp. 850-864, FOCS 1982.

[BeMi] Bellare, M., and S. Micali "How to Sign Given Any Trapdoor Function" *STOC* 88.

[BFM] Blum M., P. Feldman, and S. Micali, "Non-interactive zero-knowledge proofs and their applications," *Proceedings of the* 20th STOC, ACM, 1988.

[BDMP] Blum M., A. DeSantis, S. Micali and G. Persiano, "Non-Interactive Zero-Knowledge" *SIAM J.Comp. 91*

[BG] M. Bellare, S. Goldwasser "New Paradigms for digital signatures and Message Authentication based on Non-Interactive Zero Knowledge Proofs" Crypto 89 proceedings, pp. 194 -211

[BY] Bellare, Yung, "Certifying Cryptographic Tools: The Case of Trapdoor Permutations" CRYPTO-92 proceedings.

[DMP1] De Santis, A., S. Micali and G. Persiano, "Non-Interactive Zero Knowledge Proof Systems," *CRYPTO-87*

[DMP2] De Santis, A., S. Micali and G. Persiano, "Bounded-Interaction Zero-Knowledge proofs," *CRYPTO-88.*

[DH] W. Diffie, M. Hellman, "New directions in cryptography", *IEEE Trans. on Inf. Theory*, IT-22, pp. 644–654, 1976.

[EGM] Even S., O. Goldreich and S. Micali "On-line/Off-line Digital Signatures" *CRYPTO* 89.

[FLS] Feige, U., D. Lapidot and A. Shamir, "Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String", Proc. IEEE Symp. on Foundations of Computer Science, 1990.

[G] Goldreich O., "Two remarks Concerning the GMR Signature Scheme" MIT Tech. Report 715, 1986.

[GGM] Goldreich O., S. Goldwasser and S. Micali "How to Construct Random Functions" *JASM* V. 33 No 4. (October 1986) pp. 792-807.

[GL] Goldreich, O., and L. Levin "A Hard-Core Predicate for all One-Way Functions" Proc. 21st STOC, 1989, pp.25-32.

[GMR] S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems", *SIAM J. Comput.* 18 (1989), pp. 186-208; (also in STOC 85, pp. 291-304.)

[GoMiRi] Goldwasser, S., 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.

[GMY] Goldwasser S., S. Micali and A. Yao, "Strong Signature Schemes" *STOC* 83, pp.431-439.

[H] Hastad, J., "Pseudo-Random Generators under Uniform Assumptions", STOC 90.

[IL] R. Impagliazzo and M. Luby, "One-way Functions are Essential for Complexity-Based Cryptography" FOCS 89.

[IR] R. Impagliazzo and S. Rudich, "On the Limitations of certain One-Way Permutations", Proc. ACM Symp. on Theory of Computing, pp 44-61, 1989.

[ILL] R. Impagliazzo, R., L. Levin, and M. Luby "Pseudo-Random Generation from One-Way Functions," *STOC* 89.

[KMO] J. Kilian, S. Micali, R. Ostrovsky "Minimum Resource Zero-Knowledge Proofs", FOCS-89.

[N] M. Naor "Bit Commitment Using Pseudo-Randomness", Crypto-89.

[NY] M. Naor and M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications", STOC 89.

[OW] R. Ostrovsky, A. Wigdeson, "One-Way Functions are Essential for Non-Trivial Zero-Knowledge", preliminary draft.

[RSA] Rivest, R.L., Shamir, A. and Adleman, L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" Comm. ACM, Vol 21, No 2, 1978.

[Ro] J. Rompel "One-way Functions are Necessary and Sufficient for Secure Signatures" STOC 90.