Towards Signature-Only Signature Schemes

Adam Young[1] and Moti Yung[2]

[1] Columbia University, New York, NY, USA.
ayoung@cs.columbia.edu
[2] CertCo, New York, NY, USA.
moti@cs.columbia.edu

Abstract. We consider a problem which was stated in a request for comments made by NIST in the FIPS97 document. The question is the following: Can we have a digital signature public key infrastructure where the public (signature verification) keys cannot be abused for performing encryption? This may be applicable in the context of, say, exportable/escrow cryptography. The basic dilemma is that on the one hand, (1) to avoid framing by potentially misbehaving authorities we do not want them to ever learn the "signing keys" (e.g., Japan at some point declared a policy where signature keys may be required to be escrowed), and on the other hand (2) if we allow separate inaccessible public signature verification keys, these keys (based on trapdoor functions) can be used as "shadow public-keys," and hence can be used to encrypt data in an unrecoverable manner. Any solution within the "trapdoor function" paradigm of Diffie and Hellman does not seem to lead to a solution which will simultaneously satisfy (1) and (2).

The cryptographic community so far has paid very limited attention to the problem. In this work, we present the basic issues and suggest a possible methodology and the first scheme that may be used to solve much of the problem. Our solution takes the following steps: (1) it develops the notion of a *nested trapdoor* which our methodology is based on, (2) we implement this notion based on a novel composite "double-decker" exponentiation technique which embeds the RSA problem within it (the technique may be of independent interest), (3) we analyze carefully what can be and what cannot be achieved regarding the open problem by NIST (our analysis is balanced and points out possibilities as well as impossibilities), and (4) we give a secure signature scheme within a public key infrastructure, wherein the published public key can be used for signature verification only (if it is used for encryptions, then the authorities can decrypt the data). The security of our scheme is based on RSA. We then argue how the scheme's key cannot be abused (statically) based on an additional assumption. We also show that further leakages and subliminal leakages when the scheme is in (dynamic) use are not added substantially beyond what is always possible by a simple adversary; we call this notion *competitive leakage*. We also demonstrate such simple leaking adversary.

We hope that our initial work will stimulate further thoughts on the non-trivial issue of signature-only signatures.

T. Okamoto (Ed.): ASIACRYPT 2000, LNCS 1976, pp. 97–115, 2000.

Key words: Public Key Cryptosystems (PKCS), Public Key Infrastructure (PKI), digital signature, decryption, nested trapdoor, abuse freeness, (subliminal) leakage, design validation proofs, NIST, FIPS.

1 Introduction

Traditionally, implementations of efficient digital signature algorithms have been constructed such that the public verification keys can be made to function as encryption keys. The separation of an encryption capability from that of signature verification seems interesting in general, especially in the contexts of exportable cryptography. In fact, in any formulated policy for encryption, it seems that bodies like recovery agents have no legitimate need to be able to obtain the signing private keys of other users. But in all of the traditional systems, signing infrastructures can immediately be used as a shadow public key encryption infrastructures (a notion put forth by Kilian and Leighton [KL95]). Thus, the natural question posed by NIST, given the needs of the US federal agencies, was: "How can a public-key system incorporating both privacy as well as signatures which does not support unrecoverable encryption be designed?" This problem (the "NIST problem" in the sequel) was specifically stated in detail in a request for comments made by NIST in a revision of the DSA FIPS [FIPS97]. It read as follows:

"The Administration policy is that cryptographic keys used by Federal agencies for encryption (i.e., to protect the confidentiality of information) shall be recoverable through an agency or third-party process and that keys used for digital signature (i.e., for integrity and authentication of information) shall not be recoverable. Agencies must be able to ensure that signature keys cannot be used for encryption. Any algorithms proposed for digital signature must be able to be implemented such that they do not support encryption unless keys used for encryption are distinct from those used for signature and are recoverable."

Comments were received from various computer security companies. Yet **none** of the comments attempted to give a technical solution to the problem. The problem as stated seems quite nontrivial. It implies the existence of a public key scheme and infrastructure that can only be used for signatures and not for public key encryptions. The problem as a whole is a purely technical challenge (the need to separate escrowed encryption from signature was noted in [FY95, An97]). In this paper we attempt a solution to this problem. Note that minimizing the abuse and direct leakage of a signature scheme will make such a scheme more easily exportable, and may help prevent that which almost happened in Japan where policy makers a number of years ago attempted to escrow signature keys for law enforcement purposes.

Obviously in any on-going communication system, if one allows leakage using timing channels and other side information like correct/incorrect message structure (or other side subliminal channels) one cannot prevent leakage of information, and this information can be used cryptographically (say, for key exchanges). For such covert channels, no crypto at all is in fact needed, and just a

marking scheme (e.g. error correcting code) or a timing convention is needed. In fact, against such attacks, communication itself has to be halted forever, which is unrealistic. This is what Rivest has pointed out as the "Chaffing and Winnowing" method [R98] using authentication channels for marking. Knowing that we cannot eliminate covert channels, the harder question is "what can be done to prevent 'direct use' of the published signature verification key for encryption?" as asked by NIST. Here we also add (perhaps beyond NIST's requirements) the issue of preventing (as much as possible) **additional** direct leakages when the scheme is dynamically used. We seek to eliminate leakage channels with much larger bandwidth (and amount of work) when compared to trivial covert channels; we call this property "competitive leakage," borrowing terminology from the field of on-line algorithms.

The above issues together try to eliminate the added advantage for "abusers" resulting from the direct introduction (publishing and using) of a signature scheme. One has to understand what can and cannot be done in this area, and the answer has to be scientific and forthright. We take a first step in this direction by attempting to answer the publicly posed NIST problem.

2 Analysis of the NIST Problem

We want to prevent published keys from being abused for encryption and further prevent direct and easy use of a small number of published signatures for encryption. We note that the difficulty of completely formalizing abuse-freeness (especially in the context of signature schemes) has been noted in the literature (see Desmedt et al. [D89, BD-etal96]). We add a key registration and publication stage to a signature scheme, and we define a digital signature infrastructure that is "signature-only" as following:

Definition 1. *A Signature-Only Digital Signature Infrastructure is a digital signature algorithm (which is secure against existential forgeries under adaptive chosen plaintext attacks [GMRi88]) with the following additional properties:*

1. **Security:** *It is not possible for the CA's (together with hypothetical other authorities) to forge a signature of any user in the system (even an existential forgery attack).*
2. **Published Key misuse freeness:** *The public verification key cannot be used to encrypt data in an unrecoverable manner.*
3. **Published Key leakage resistance:** *The public verification key has a very small subliminal channel (e.g., < 32 bits or so), and thus cannot be used to effectively display a complete shadow public key.*
4. **Signature leakage resistance:** *The signatures created using the private signing key are shadow public key resistant. Namely, it is not possible to use the properties of a given signature per se, to derive a shadow public-key. In addition we can strengthen the requirement and demand competitive leakage: the number of signatures (and complexity of derivation) required to get a shadow public key is approximately the same as in a trivially leaking channel (available by the communication).*

The first property above strengthens the traditional security and requires it to hold against the certification authorities (CA) which may hold extra information. The second property deals with abuse as required by NIST, whereas the other properties deal with add-on leakage of the system (compared to a non cryptographic system), which we have added to NIST's original requirement. We note that we do not attempt to eliminate subliminal leakage, but rather try to minimize add-on leakage as compared to existing trivial channels. In fact, we will show that every on-going signing leads to trivial (unavoidable) leakage.

Additional related work: The notion of a subliminal channel has been suggested by Gus Simmons [Si85]. A subliminal channel in a cryptosystem is a channel that allows a user of the system to leak information of his or her choosing through that channel. Simmons showed that information can be leaked in digital signatures, including DSA signatures [Si93]. It is known that in many operational environments, covert channels inherently exist. Within the context of key recovery systems, Kilian and Leighton showed that subliminal channels can be exploited to display unescrowed public keys within channels that exist in escrowed public keys, which are displayed by the CA [KL95]. They call this form of attack which easily adds a public key directory, shadow public key abuse. Abuses of cryptosystems have been discussed by Desmedt [D89]. It was originally believed (by some) that the NSA designed DSA [DSS91] could not be used for public key encryption or key exchange operations. Yet it was shown how the DSA verification key can be used for encryption (overtly [NR94] and even covertly within a single signature [YY97]). Such capabilities cannot be permitted if one is to design a scheme solving the NIST problem.

Methodology: In our methodology, we first discuss the security of the scheme (reducing from a known problem), we then show (under a new assumption) how the public keys in the scheme we present cannot be used as trapdoors in known systems in the sense that, if they are used, then the authorities are able to recover data encrypted with them. We then present methods that limit additional subliminal leakage of information. We do not employ cumbersome interactive techniques (e.g., the schemes in [D89, KL95]), but rather we employ non-interactive methods that assure that near-random choices of public parameters are made (making random choices via the use of one-way hash functions which are regarded as random oracles– an idea rooted in the DSA parameter generation procedure). We also guarantee leakage-freeness in the signatures themselves by using deterministic (and pseudorandomized) signing algorithms rather than randomized signing algorithms.

To claim security, we reduce the ability to forge (by the CA which holds more information than the signature verifying users) to the ability to break RSA as a one-way function (in the random oracle model). We assume RSA is a one-way function:

Security Assumption: Given $n \in Z^+$ s.t. $n = pq$ where p and q are distinct random primes of size k, and a fixed $e \in Z^+$ s.t. $gcd(e, \phi(n)) = 1$, and given a random $c \in Z_n^*$, finding $m \in Z_n^*$ s.t. $m^e = c \pmod{n}$ is hard (i.e., for any polynomial it takes more than polynomial in k time, for all k large enough).

Naturally, we need to claim that a public key cannot be used for public key encryptions. Since public key information is available, we **must** assume or show the non-existence of a (probabilistic) poly-time algorithm that uses the public key as it was created to encrypt data without the CA being able to decrypt that data. This is how the notion of a "nested trapdoor" helps: on one hand the CA cannot sign (since the CA does not know the innermost trapdoor or otherwise is in violation of the security assumption), but on the other hand, the CA can decrypt based on the information it holds (using the middle trapdoor).

For the outermost trapdoor, we now give the new assumption about the public key information that forms the basis for our proposed scheme. It is needed, given the state of the art of subliminal freeness in signature schemes. It basically states (based on the state of the art) that once the factorization of a randomly generated specially structured number is known (and its structure proven), it cannot serve as a public encryption key. More precisely:

No-Abuse Assumption: There does not exist a secure public key encryption algorithm that takes n as the input public key, where n (of size k) is the product of two large known (to the CA only, in our case) primes p and q with $p = 2tp_1q_1 + 1$ and $q = 2q_2 + 1$, such that p_1 and q_1 are secret primes, t is a small public safe prime, and q_2 is prime (and given e a fixed number of size polynomial in k, such that $gcd(e, \phi(\phi(n))) = 1$).

Note that even though p_1q_1 is known (to the CA only), the encryption algorithm does not know it since it only takes n (and e) as input. Given the state of the art, the above assumption is mathematically plausible (its validation is an interesting issue, and knowing whether a weaker assumption suffices is left open). The plausibility of the assumption can be supported as follows:

- The assumption's public refutation can be done by proposing an encryption scheme which bypasses the above factorization knowledge (of the CA in our case) by using either (1) an old trapdoor, or (2) a new one. The latter, in fact, would be an interesting new public key suggestion.
- If a more general version of the assumption (not restricting to the exact structure of n above but using generic n) does not hold, then it implies the existence of a shadow public key attack on **any** factoring based key escrow system where w.l.o.g. $p - 1$ is hard to factor.

Main Idea: Now that we discussed the assumptions, let us describe briefly the main idea behind the scheme. The value $\phi(\phi((n))$ effectively constitutes the private signing key of the user, $Z_{\phi(n)}$ is the domain in which the signatures are computed, and Z_n is the domain in which signature verification is performed. Our scheme is setup such that neither $\phi(n)$ nor $\phi(\phi(n))$ is known to the verifier, and such that only knowledge of n is needed to verify. Also, we insure that $\phi(n)$ *is* known to the CA. Hence, any PKCS predicated on the difficulty of factoring n provides no security against the CA. So, in our scheme there is "double-decker" exponentiation. The upper most deck is known only to the signer, the middle deck is known to the CA and the signer, and the bottom deck is known to everyone. We call this a nested trapdoor construction.

Note that double-decker techniques based on the discrete log problem have been utilized before [St96, CaSt97, YY98]. However, the novel double-decker application we present requires the use of composites and is predicated on the problem of factoring.

Remark: Naturally, any infrastructure that provides authenticity can be abused via leakage by criminals to permit *authenticated* communications which are untappable (since there are non cryptographic means to leak data). This is irrelevant since the problem posted by NIST appears to deal with direct abuse of signatures for encryption (obviously, NIST did not overlook the possibility of subliminal/covert channel use and direct "key exchanges"). For instance, the certified signature keys can be used to conduct an *authenticated* key exchange. When criminals are given the ability to authenticate themselves then this type of abuse is entirely unavoidable. **Hence, the property that we require is that criminals be forced to go outside of the provided key recovery/authentication mechanisms to conduct the untappable authenticated communications** (e.g., force criminals to do a key exchange, which they can authenticate if they so choose). What is achieved here, in fact, is that under our assumptions and in our proposed system, a user *cannot* abuse keys directly, and usage of the system competitively, to conduct secure communication. Recall that by 'competitively' we mean that we do not allow substantial increase in leakage via the keys or the signatures (requirements beyond NIST's original problem). The scheme can therefore be viewed as the first unescrowed digital signature scheme which, as much as possible, does not defeat the purpose of a recoverable PKI when added to a recoverable PKI on its outset (but requires extra work from abusers). We note that in general, in the area of abusing escrow encryption, the malicious adversaries can always bypass the system and the remaining scientific challenges are to protect against more benign adversaries which do not invest much extra effort (e.g., they use the available keys) or when this bypassing is to be revealed by other means in the communication system.

3 Re–examination of Existing Signature Schemes

In this section we will analyze the shortcomings of various digital signature algorithms in light of the NIST requirement. None of the schemes were designed with NIST's problem in mind, so these are not weaknesses but rather a reflection of the state of the art.

RSA: We take the opportunity to look at an RSA variant which makes the RSA signing function a uniform trapdoor permutation via pre-hashing. In this system, n is the product of two large primes p and q, and e is a public value such that $gcd(e,(p-1)(q-1)) = 1$. e and n are the user's public verification keys, and the inverse of e mod $\phi(n)$ is d, the user's private signing key. The signature on m is $s = H(i||m)^d \ mod \ n$, where i is the smallest positive integer making $H(i||m) \ mod \ n \ \in \ Z_n^*$ (as in [BR93]). To verify a signature we check that $s^e \ mod \ n = H(i||m)$. Clearly (e,n) can be used to encrypt data. If d, p, or q is given to the authorities, it follows that the authorities (CA) can forge signatures.

This applies to Rabin [Ra78] and every factoring based scheme, including Esign where $n = p^2q$.

ElGamal and Relatives: In ElGamal [ElG85], the public key is $y = g^x \ mod \ p$, where x is the private signing key. Here g is a public generator modulo the public prime p. If the private key x is not given to the authorities, y can be used for encryption (e.g., the ElGamal encryption scheme). If x is given to the authorities, obviously they can forge signatures. The same holds for DSA, Elliptic Curve DSA, Schnorr, undeniable signatures, Nyberg and Rueppel, etc.

Fiat-Shamir: In Fiat-Shamir, n is the product of two primes and none of the users know the factorization of n. To generate a public key, a user generates k different quadratic residues v_1, v_2,...,v_k modulo n. This vector is the public key. The scheme therefore succumbs to the following shadow public key attack. The user chooses a random x and squares it to get v_1. Let $g = v_1$ be a generator of hopefully a large subgroup containing all quadratic residues in Z_n^*. To generate v_2, a malicious user chooses w at random and sets $v_2 = g^{2w} \ mod \ n$. Thus, v_1 and v_2 are quadratic residues, and constitute a shadow public key for generalized ElGamal mod n. The shadow private key is $2w$. (Even, if we only have v_2, v_1 can be derived from n and the user's name, or be a constant)

Okamoto '92: To date, no large subliminal channel in this scheme is known. Recall that in this scheme, the verification key is $v = {g_1}^{-s_1} {g_2}^{-s_2} \ mod \ p$. Here g_1 and g_2 have order q modulo the public prime p. The values for g_1, g_2, and q are public. The private key is (s_1, s_2). Both s_1 and s_2 are chosen randomly modulo q. Okamoto is based on the representation problem modulo p (and so are fail-stop signatures).

To sign a message m in Okamoto, we choose $r_1, r_2 \in_R Z_q$. We then compute $e = H({g_1}^{r_1} {g_2}^{r_2} \ mod \ p, m)$. Here H is a one-way hash function. We then compute $y_1 = r_1 + es_1 \ mod \ q$ and $y_2 = r_2 + es_2 \ mod \ q$. The signature on m is the triple (e, y_1, y_2). To verify the signature we check that $e = H({g_1}^{y_1} {g_2}^{y_2} v^e \ mod \ p, m)$.

At first sight it seems as if the scheme is a good candidate for a solution. For, suppose that we don't give to the authorities s_1 and s_2. Then the authorities can't forge signatures. But, then we need to insure that v cannot be used as a shadow public key. Suppose that v can be used as a public key in a public key encryption algorithm. Since Okamoto is extendible to three or more bases, maybe there is no encryption algorithm if the representation uses three bases, or four bases, etc. This line of reasoning begs the question as to whether or not there exists a 'generalized Okamoto' public key encryption algorithm.

We will now answer this question in the affirmative. Indeed, there is a public key algorithm that uses public keys based on the representation problem with any number of bases. We will demonstrate this algorithm where two bases are used. To public key encrypt a message m using v as in Okamoto's scheme, we do the following:

1. $k \in_R Z_q$, $a = g_1^k \ mod \ p$, $b = v^k \ mod \ p$, $c = {g_2}^k m \ mod \ p$
2. The ciphertext of m is (a, b, c)

To decrypt we compute:

1. $a' = a^{-s_1} \ mod \ p$ which equals ${g_1}^{-s_1 k} \ mod \ p$
2. $b' = b/a' \ mod \ p$ which equals ${g_2}^{-s_2 k} \ mod \ p$
3. $m = c/({b'}^{-{s_2}^{-1}}) \ mod \ p$

This algorithm can be easily extended to handle representations using more bases. The ciphertext is an $(m+1)$-tuple if m bases are used in the representation of v. Thus, this scheme and it's extentions using more bases does not meet the requirements.

Naor Yung signature scheme: A digital signature algorithm (which is presented as an important plausibility result, not as an efficient scheme) was presented in [NY89] that is based on the existence of one-way permutations (Rompel uses this exact setting but with any one-way function). The system is provably secure against adaptive chosen message attacks. It operates by having each user publish a row of windows, where each window is a pair of values that form commitments of the user via a one-way permutation value. Since the scheme (and its follow up work) is not based on trapdoor functions it departs from the Diffie-Hellman paradigm. Does this mean it is shadow public key resistant? No, the problem is that even though the system *itself* does not employ a trapdoor function, it could display one subliminally assuming trapdoor functions exist. The problem is that each window can leak several (say, 20) subliminal bits, thus allowing the system to directly leak a shadow public key (e.g., an RSA public key which is not held by the authorities).

4 The Signature Scheme

In this section we describe our construction which is a first answer to the NIST problem. We present the scheme and the user registration procedure.

4.1 Key Generation

Let e be an odd value (fixed and bounded as discussed below). Let M be a security parameter (which is say, a power of 2). Let p_1 and q_1 be $M/2$ bit primes. Let $q = 2q_2 + 1$ be a safe prime. These primes adhere to the following constraints for proper system operation:

1. Each of $p_1 - 1$, $q_1 - 1$, and $q_2 - 1$ have a large prime in their factorization.
2. $\gcd(e, (p_1 - 1)(q_1 - 1)(q_2 - 1)) = 1$.
3. There exists an M_1-bit (e.g., $M_1 = 63$) safe prime t s.t. $p = 2tp_1q_1 + 1$ is prime[1] and s.t. $\gcd(e, \phi(t)) = 1$.

[1] An alternate implementation chooses $(q-1)/2$ to have the same form as $(p-1)/2$, but $(q-1)/2$ has a different factorization.

If we want $|pq|$ to be a power of 2, we can choose q to be $M - M_1 - 1$ bits long. We incorporate t into $p - 1$ to make key generation fast. To provide protection against shadow public key abuse (see [KL95] for the attack on composites), the following additional constraints are needed to reduce subliminal leakage:

1. s and s' are chosen (randomly).
2. $H_1(s)$ (or $H_1(s) + 1$, see below) is the same as the upper half of the bits in the bit representation of $p_1 q_1$.
3. q is chosen by computing $q = H(s')$ and testing for primality and strong primality (more elaborate methods are possible, e.g., the method for generating parameters for DSA).

To accomplish step 2, an algorithm similar to [L98, YY96] can be employed. Thus, either $H_1(s)$ is the upper order bits, or $H_1(s) + 1$ is the upper order bits due to a borrow bit being taken. Here H_1 and H are suitable (ideal) one-way hash functions. This step is to avoid the leakage of $M/2$ bits in the composite $p_1 q_1$. The values p, q, and t are found to satisfy the above. The key generation algorithm then performs the following computations:

1. $n_1 = 2tp_1q_1q_2$
2. Compute the smallest value s'' that makes $g' = H_2(s, s', s'', n)$ a generator mod p, $g'' = H_3(s, s', s'', n)$ a generator mod q, and $\gcd(x_i, n_1) = 1$, where $x_i = h_i'(s, s', s'', n, i)$ for $i = 1, ..., K$ (K odd), and H_2, H_3, h_i' being appropriate ideal hash functions. We insist that s'' is, say, at most 24 bits in length.
3. Chinese remainder $g \equiv g' \ mod \ p$ with $g \equiv g'' \ mod \ q$ to get $g \ mod \ pq$ (g then has order $\lambda(pq)$).
4. $n = pq$
5. Compute $g_i = g^{x_i} \ mod \ n$ for $i = 1, ..., K$
6. $d = e^{-1} \ mod \ \phi(\phi(n))$ (We will use ϕ^2 to denote $\phi(\phi())$.)
7. Compute T to be a NIZK proof of knowledge of the factorization of $(p-1)/2t$ into two distinct primes (in the random oracle model). A modification of the techniques of [GHY89] and [BFL91] enables this (the short proof of [PS00] may apply as well). T can also be an interactive ZK proof if we allow interaction.

Note that $n_1 = \lambda(n)$ is the Carmichael function λ of n. The public verification key is $(g, g_1, ..., g_K, e, n)$. The private signing key of the user is $(x_1, x_2, ..., x_K, d, n_1)$. To register the public verification key with the CA, the user sends to the CA the tuple $(s, s', s'', p, q, t, e, T)$. The value for s'' must be sent, to assure the CA with very high probability that g' generates Z_p. In practice n is about twice the size of a regular RSA modulus.

It is obviously imperative that the g, g_i's and n, when taken together, cannot easily allow n to be factored. Otherwise, the composite $\lambda(n)$ would be available to all users and could be used as a shadow public key. Since random g, g_i's are samplable, the following fact holds:

Fact 1 *The values for the g, g_i's and n, when taken together, cannot be used to factor n.*

4.2 CA Registration

The CA receives $(s, s', s'', p, q, t, e, T)$. The CA computes v_1 to be $(p-1)/2t$. The CA then sets z to be the upper half of the bit representation of v_1. The CA computes n, g', g'', g, and the g_i's in the same way as the user. The CA verifies all of the following things:

1. p is prime, t is prime, $t \mid p-1$,$|p-q|$ is large, g'' generates Z_q, etc.
2. $H_1(s)$ or $H_1(s)+1$ equals z.
3. $g^{(p-1)/2}, g^{(p-1)/t}, g^{(p-1)/v_1} \neq 1 \ mod\ p$.
4. checks that T is valid.

T convinces the CA that the user knows the signing private key. T and step 1 proves that there are 16 possible orders for g'. T and step 3 proves that $ord_p(g') \in \{2tp_1, 2tq_1, 2tp_1q_1\}$, i.e., it can be only 3 of those 16. Recall that if p is prime and $w \mid p-1$, then the number of least residues mod p with order w is $\phi(w)$. Since g' is chosen by an oracle, it has order $2tp_1$ with probability $(t-1)(p_1-1)/p$, which is guaranteed to be negligible since v_1 must be made difficult to factor. By making p large enough, this quantity is still negligible even after a polynomial number of oracle queries by a cheating prover. The same holds for $2tq_1$. So, the order of g' mod p is $2tp_1q_1$ with overwhelming probability. Anyway, recently an efficient method to prove in ZK that a number is maximal order was shown [JY00]. Clearly g has order $\lambda(n)$ if g' and g'' are generators. Similarly, the CA verifies that the order of g_i's (for $i = 1, ..., K$) are the same as the order of g (i.e., that $\gcd(x_i, n_1) = 1$).

If all of the verifications pass then the CA publishes (and certifies) the string $(g, g_1, ..., g_K, e, n)$ as the user's public verification key.

4.3 Signing and Verifying Messages

Let $a||b$ denote the concatenation of string a with string b, let h_i's be ideal full domain hash functions into $\{1, 3, 5, ..., (n+1)/2 - 1\}$ (a random oracle answer concatenated with a 1 as the least significant bit). To sign an arbitrary message m, the user computes $c = (x_1h_1(j||m) + x_2h_2(j||m) + ... + x_Kh_k(j||m))^d \ mod\ n_1$, where $j > 0$ is the smallest integer making all $h_j(j||m)\ mod\ n_1 \in Z^*_{n_1}$ (as required by the full domain hash method [BR93], and will be 1 almost always, thus for brevity we omit the notation $j||m$ and use m instead). The signature on m is c. Note that the signer can use the Chinese Remaindering method to make signing faster. The verifier checks that:

1. $c < n/2$
2. ${g_1}^{h_1(m)} * {g_2}^{h_2(m)} * ... * {g_K}^{h_K(m)} \ (mod\ n) = g^{c^e} \ (mod\ n)$

If (1) and (2) are not satisfied then the signature is rejected. To substantiate verification (1), we need to show that $\lambda(n) < n/2$ for all properly chosen n's. This can be seen from the following.

$$\lambda(n) = lcm(p-1, q-1) = lcm(2tp_1q_1, 2q_2) = 2tp_1q_1q_2$$
$$n/2 = (1/2)(4tp_1q_1q_2 + 2tp_1q_1 + 2q_2 + 1) > 2tp_1q_1q_2$$

So, the claim holds. The reason we insist on making signatures less than $n/2$ is to avoid subliminal leakage of information in signatures. Below we elaborate more on this.

Note that since $e > 2$ the only way to verify the signature is to determine $g^{c^e} \ mod \ n$ by exponentiating e times:

$$((g^c \ mod \ n)^c \ mod \ n)^c ... \ mod \ n$$

This is necessary because the modulus in the exponent is unknown to the verifier, thus the verifier cannot first compute $c^e \ mod \ n_1$ and then use this as the exponent for g. Note that signing is done in the upper most deck with the signature value in the middle-deck, and verification is performed in the bottom deck. The system therefore utilizes composite-based double-decker exponentiation. For efficiency we assume e to be small (theoretically, e which is polynomial in the size of n is still feasible).

Proposition 2. *Signature verification is complete (i.e., properly generated keys and signatures will always verify).*

Proof. If n is generated properly, then verification (1) will always pass. Now consider verification (2). One can verify that $c^e = x_1h_1(m) + x_2h_2(m) + ... + x_Kh_K(m) \ mod \ \lambda(n)$. Since g and the g_i's have order $\lambda(n)$, by the construction of the g_i's the claim is proved. □

5 Security Against Forgeries: Validation Proof

Recall that the user sends to the CA the tuple $(s, s', s'', p, q, t, e, T)$. We need to show that given this information, and given a polynomial number of message/signature pairs, it is not possible for the CA to forge messages (i.e., that the CA can't choose messages and sign them, or even produce existential forgeries). We will do this in two steps. First we will show that the tuple that is given to the CA during certification does not help the CA forge, then we will show that a slight modification of our system is identical in security to RSA, and that this modification implies that our scheme is also secure. We will then show even further that in the random oracle model, our system is secure against adaptive chosen message attacks (we are aware of the advantages of having such proofs, e.g. [BR93], as well as the existence of limitations for certain implementation constructs which are not applied here [CGH98]).

Note that the CA knows $\lambda(n)$, since $\lambda(n) = n_1 = (1/2)(p-1)(q-1)$. The CA does not know $\phi(\phi(n))$, since the CA doesn't know the factorization of p_1q_1. We need to show that no information about the factorization of p_1q_1 is leaked to the CA by $(s, s', s'', p, q, t, e, T)$, excluding deliberate leakage (i.e., assuming

honest user). First, nothing is leaked by T since this is a zero-knowledge proof. Also, t need not even be sent by the user, since it can be found easily by the CA. The seeds s, s', and s'' contribute the information constituting the upper order bits of p_1q_1, the prime q, and the generators g', g'', and the exponents bringing g to g_i where $i = 1, ..., K$, and nothing else, since they constitute the preimages under the random oracle hash functions H_1, H_2, and H_3, and $h_1^{'}, ..., h_K^{'}$. To see this, note that under the random oracle assumption, each hash value is chosen uniformly at random from its respective set, and the result is therefore independent (as viewed by all poly–time algorithms) of the random hash function's input. It follows that the only information leaked by s, s', and s'' is the upper order bits of p_1q_1 (which is already known to the CA from p), and also q, g',g'', g, and $g_1, ..., g_K$. If we can show that no information about the factorization of p_1q_1 is leaked by g' then we will be done.

Suppose that given a generator g' of Z_p, and given p, an oracle A can be used to factor $p-1$, where p is prime. In other words, $A(g', p)$ outputs the factorization of $p-1$. Then A can be used to factor any composite n as follows. Choose a prime k randomly s.t. $2kn+1$ is a prime number p'. Choose a value $h \in_R Z_{p'}$ and run $A(h, p')$. If A fails, choose (k, h) over and repeat. Since there are $\phi(p'-1)$ generators of $Z_{p'}$, we expect to factor n with probability $\phi(p'-1)/(p'-1)$ each time we invoke A. Thus, $p-1$ will be factored in expected poly–time. We have shown via random reduction that factoring n is no harder than implementing the oracle A. So, implementing the oracle A which uses g' and p to factor $p-1$ is no easier than factoring n. Since g' is chosen (accessed) almost randomly , it follows that g' contributes in no way to the CA's ability to factor $(p-1)/2t$. Thus, we have shown that:

Lemma 3. *No information about the factorization of p_1q_1 (and hence $\phi^2(n)$) is leaked to the CA by $(s, s', s'', p, q, t, e, T)$.*

To forge, the CA need not necessarily output a c satisfying $c < \lambda(n)$, since the users who will perform verifications do not know $\lambda(n) = n_1$. The CA therefore needs only to output a $c < n/2$ s.t. $g^{x_1h_1(m)+x_2h_2(m)+...+x_Kh_K(m)} \ (mod\ n) = g^{c^e} \ (mod\ n)$ to forge, since users can be certain that c must be less than $n/2$. But, if the CA knows a valid $c > n_1$, then the CA knows a valid signature in the correct range, since $c \ mod\ n_1 < n_1$. Thus, a CA can output an acceptable forgery implies that it knows a forgery in the correct range. The following proves that forgeries are possible in our system by the CA iff the CA can break RSA.

Theorem 4. *The CA can forge signatures of users in our system (without hashing) using exponent e iff the CA can forge signatures in the RSA scheme (without hashing) using exponent e.*

Proof. Let $n = pq$, and let e be a small number s.t. $gcd(e, \phi(\phi(n))) = 1$. Furthermore, let t be a safe prime, and let $p = 2tp_1q_1 + 1$, $q = 2q_2 + 1$, and q_2 be prime. Suppose that the CA has an oracle A_e that uses an exponent e such that when given (n, m) produces $c < n/2$ s.t. $g^{x_1m+...+x_Km} \ mod\ n = g^{c^e} \ mod\ n$.

In other words, $c = A_e(n, m) < n/2$ is a valid signature on m in our algorithm (without hashing), hence the CA can forge signatures in our system. Let n' be an RSA modulus, i.e., $n' = p_3 q_3$ where p_3 and q_3 are prime. Let the RSA exponent be the same e as in our scheme. Thus, $gcd(e, \phi(n')) = 1$. Let d' be s.t. $ed' + w\phi(n') = 1$ for some integer w. The CA can use A_e to forge RSA signatures for the public key (e, n') by choosing a small safe prime t randomly s.t. $p' = 2tn' + 1$ is prime, and by choosing a random safe prime q' where $q' = 2q'' + 1$. The CA makes sure that $gcd(e, \phi(t)\phi(q'')) = 1$. Let d'' be s.t. $ed'' = 1 \ mod \ \phi(2tp_3q_3q'')$. To forge an RSA signature, the CA chooses $r \in_R Z^*_{\lambda(p'q')}$ and computes $c' = r^{-1}A_e(p'q', r^e m/(x_1 + ... + x_K)) \ mod \ 2tn'q''$. It follows that $c' = m^{d''} \ mod \ 2tn'q''$. The CA then sets $c = c' \ mod \ n'$.

$$c' = m^{d''} + k2tp_3q_3q'' \text{ for some integer } k$$

$$c = c' \ mod \ p_3q_3 = m^{d'' \ mod \ \lambda(p_3q_3)} \ mod \ p_3q_3$$

But, $d'' \ mod \ \lambda(p_3q_3)$ is the inverse of e modulo $\lambda(p_3q_3)$. To see this note that,

$$ed'' + v\lambda(2tp_3q_3q'') = 1 \text{ for some integer } v$$

But, $\lambda(p_3q_3) \mid \lambda(2tp_3q_3q'')$. So, $e(d'' \ mod \ \lambda(p_3q_3)) = 1 \ mod \ \lambda(p_3q_3)$. Thus, c is a valid signature on m in RSA mod p_3q_3.

Conversely, suppose that the CA can forge signatures in the RSA scheme using the exponent e. So, assume that the CA has an oracle B_e that uses e, a composite of two different primes n_1 and a message m, that produces a valid RSA signature $c \ mod \ n_1$ on m. That is, $c = B_e(n_1, m)$ is a valid RSA signature on m. A CA knowing the composite p_1q_1 and the prime q_2 of a user can use B_e as an oracle to forge in our system as follows. The CA computes $c' = B_e(p_1q_1, r^e m(x_1 + ... + x_K))/r \ mod \ p_1q_1$ where r is chosen randomly from its respective message space, and computes $c'' = (m(x_1 + ... + x_K))^{e^{-1}} \ mod \ 2tq_2$. Note that it was verified by the CA that $gcd(e, \phi(t)) = 1$, $p = 2tp_1q_1 + 1$ is prime, and that $q = 2q_2 + 1$ is a safe prime. The CA Chinese remainders $c' \ mod \ p_1q_1$ with $c'' \ mod \ 2tq_2$ to produce $c \ mod \ 2tp_1q_1q_2$. □

Note that the above proof constitutes a randomized reduction where if oracle A_e exists, then we show the existence of only one other oracle, namely oracle B_e, and vice-versa (i.e., existence of A_e does not necessarily imply the existence of $B_{e'}$ for all e'). However, we need only assume that every oracle succeeds on a fixed fraction of the message space. Thus, our system is no more or less secure than RSA. Since the above holds for the CA, clearly *the same holds for all of the users* in the system, who have access to even less information than the CA.

We will now prove that our signature scheme is secure against adaptive chosen message attacks under the random oracle model, assuming that RSA is secure. Since t and q_2 are known to the CA, the CA can always compute the partial signature $c_1 = (x_1h_1(m) + x_2h_2(m) + ... + x_Kh_K(m))^{e^{-1} \ mod \ \phi(tq_2)} \ mod \ 2tq_2$ on a message m. Therefore, when the user sends the CA a signature c on message m, the user can send the signatures c_1 and $c_2 = (x_1h_1(m) + x_2h_2(m) +$

$... + x_K h_K(m)^{e^{-1} \ mod \ \lambda(p_1 q_1)} \ mod \ p_1 q_1$ instead, since the CA can just Chinese Remainder c_1 and c_2 to get c. For the purposes of this proof, we will assume a digital signature scheme in which everyone knows p, q, t, g', g'', and the g_i's and in which the two signatures (c_1,c_2) constitute the actual signature on m. The generation and verification of c_1 is straightforward, and the generation and verification of c_2 is as in RSA with hashing. Though it is important to note that this proof holds for the specific full domain hash function $Hash(m) = x_1 h_1(m) + x_2 h_2(m) + ... + x_K h_K(m)$. Now, since the basic h_i are random oracles over their domain (of odd numbers), and adding mod n_1 assures that we stay within the domain (provided K is odd), we can use $Hash$ as an ideal (random oracle) hash function (over a certain domain), and the arguments of the security of the full domain hash hold.

Theorem 5. *Assuming the random oracle assumption holds on the h_i's, and assuming that our security assumption holds (i.e., RSA being a one-way function), the signatures c_2 are secure against adaptive chosen message attacks with respect to the CA.*

Proof. We apply the Full-Domain Hash analyzed in [BR93, C00] based on a random oracle over a sub-domain of $Z^*_{n_1} = Z^*_{2tp_1q_1q_2}$ (explained below), and get a full domain hash $Hash$, which takes messages m and gives a random element from that full domain. Clearly, c_1 provides no assurance that m is authentic, since c_1 can be produced by anyone with the CA's knowledge. It follows that our digital signature scheme is secure against adaptive chosen message attacks iff the key generation algorithm for c_2, the signing algorithm for c_2, and the verification algorithm for c_2 is secure against adaptive chosen message attacks. It was shown in [BR93, C00] that indeed these algorithms (using a full-domain hash) are secure against adaptive chosen message attacks. Now, using the Chinese Remainder Theorem, $c_2 = (x_1 h_1(m) + x_2 h_2(m) + ... + x_K h_K(m))^{e^{-1}} \ mod \ p_1 q_1$ In the security proof the value of c_2 before the exponentiation (namely, $x_1 h_1(m) + x_2 h_2(m) + ... + x_K h_K(m)$ is in fact assumed to be chosen by a random oracle drawing elements from $Z^*_{p_1 q_1}$, which is the domain over which we want to prove security. This can be assumed due to the 1-1 mapping between the elements assured to be (almost always) in $Z^*_{n_1}$ and the set of their remainders: c_1, c_2. Furthermore, due to the random oracle assumption which gives us a degree of freedom in fixing polynomially many values (of c_2) in the attack protocol, when arguing about a value, we can always solve for the randomly chosen c_2 element of our choice (by "playing" with the actual value of one of the basic hash values h_i) and then, in the simulation, having a desired challenged value or probed element to be a random element of our choice in $Z^*_{p_1 q_1}$ (which we already know the result of applying the exponentiation to). By doing so we determined c_1 with an arbitrary value, due to the algebraic relation. Under the above proper random oracle assumption, we can turn the signature values (c_2) into desired (random) challenges whose inverses are actually accessible anyway (by our choice), and then "chosen plaintext" which is the input to the RSA operation is reduced to a "random plaintext" attack on the RSA operation (namely, it reduces the

adaptively chosen plaintext forgery attack to the security of RSA as a one-way function into the domain of the c_2 elements). The full domain hash assumption enables us to argue that the "altered random oracle" (which is modified in only a few polynomially-many places) is statistically indistinguishable from the original one. A more sophisticated oracle answering strategy is in [C00]. □

6 Arguing Published Key Abuse Freeness

To show the resistance of $(g, g_1, ..., g_K, e, n)$ to abuses, it is necessary to show two things:

1. Key misuse freeness: the values in $\{g, g_1, ..., g_K, e, n\}$ cannot be used directly as a shadow public key in any PKCS.
2. Key leakage resistance: the information in $(g, g_1, ..., g_K, e, n)$ cannot display enough subliminal bits to constitute a secure shadow public key.

To show (1), first notice that we claim that n cannot be used for unrecoverable RSA/factoring-based encryptions using n as the composite modulus, simply because the CA knows the factorization of n. From the No-Abuse assumption there is no other way to exploit n (here is where we strongly employ this assumption).

Next observe that the way g is generated, makes it random maximal order element. Assuming discrete logarithm is a hard problem on random instances, even when the factorization is known (this is a regular discrete log assumption), will prevent the user from using it as a base for a public key. As for g_i's, the CA is aware of the same information about them as the user does. Finally, since small enough e can be fixed among all the users, it cannot be or contribute to a shadow public key.

We will now observe a simple yet strong result which proves about the system containing (n, e) the following:which uses a fixed e (without the g and the g_i's):

Theorem 6. *If there exists a shadow public key attack on pq where pq "is" the shadow public key then there exists a shadow public key attack on any composite pq based key escrow system where the user chooses p and q at the users own discretion and where p (or q) is escrowed.*

This directly follows from the fact that what is published in this modified system is exactly what is published in any composite pq based key escrow system (where users can choose the form of, w.l.o.g., $p - 1$).

Next we argue (2), namely that there is no key leakage (based on the current state of the art). We need to show that there is no subliminal channel in n of significant bandwidth. Recall that $\lambda(n)$ has p_1q_1 and q_2 in its factorization. The value p_1q_1 was generated in such a way as to minimize subliminal channels in it. This was done by making its upper order bits a one-way hash (behaving like a random oracle) of a random number s, and is precisely the method used to foil high bandwidth shadow public key attacks in RSA moduli (by filling the

subliminal channel). By forming p_1q_1 in this way, there can only be a small (logarithmic length) subliminal channel in p_1q_1 (we may also insist that the values chosen via the one way function have a fixed number of bits). Similarly, q_2 can only have a similarly small subliminal channel since it is chosen by a random oracle. Since t is a small odd value, it provides no significant subliminal channel either. It follows that $\lambda(n)$ is mostly random and has no substantial subliminal channel, so neither does n (which is built from it in a given format).

Now consider g. The value g is computed based on g' and g''. But, g' is chosen by a random oracle whose input are the values s, s', n, and s_1. So, g' has no substantial subliminal channel. We make s, s', and n inputs to a random oracle and increment s'' from zero to find g' as a precautionary measure to limit the amount of subliminal leakage in g. This precaution works because s'' is small, and thus a malicious user is forced to regenerate n if the user fails to leak enough information in g by the time s_1 reaches its maximum allowable value. Thus, the key generation time is inhibited when key generation is modified to leak subliminal information via g. The same argument applies to g'', so g is mostly random and has no substantial subliminal channel. In fact, g is effectively chosen almost uniformly at random (by a random oracle). Furthermore g_i is chosen via x_i and g, and x_i is formed from the same information (under a different random oracle) as g' and g''. We note that using one-way hashing to generate trap-door free parameters was first proposed by NIST with respect to DSA. This concludes the arguments suggesting that $(g, g_1, ..., g_K, e, n)$ does not leak enough subliminal information. In practice it may be possible to leak on the order of O(log log(n)) bits or so if an attacker works hard (searching for a pattern), but again, we are protecting only in the relative sense: it may be that so little amount of information can be subliminally sent outside the cryptographic system.

7 Arguing Signature Leakage Resistance

Can a single signature leak information? Clearly, if there were a way to output more than one signature for a given m, it may be possible to leak bits of subliminal information. So, we would like to show that for each message m, with overwhelming probability there is only one valid signature c on m. Then, since the signing algorithm is deterministic, we will be well on our way to showing that our signatures are leakage resistant from this aspect.

Proposition 7. *With overwhelming probability, it is not possible to find an m s.t. $c, c' < n/2$ will both be accepted as valid signatures on m.*

Proof. From signature verification it is clear that all the valid signatures on m are specified by

$$c_j = j\lambda(n) + (x_1h_1(m) + ... + x_Kh_K(m))^d \ mod\ \lambda(n)) < n/2, \text{ for } j \geq 0$$

Clearly c_0 will always be a valid signature. If h_j is a random oracle, then for each m, the probability that c_j is a valid signature is negligible if $j \geq 1$. We will

prove this by first proving that it holds for $j = 1$, and the fact that it holds for all $j > 1$ will be immediate. Note that if $j = 1$ then c_j is a valid signature iff $(x_1h_1(m)+...+x_Kh_K(m))^d \ mod \ \lambda(n)) < n/2-\lambda(n) = tp_1q_1+q_2+1/2$ Since the h_j's are random oracles, then $(x_1h_1(m) + ... + x_Kh_K(m))^d \ mod \ \lambda(n)$ is chosen randomly from $Z_{\lambda(n)}$, independent of m. Thus, $(x_1h_1(m) + ... + x_Kh_K(m))^d \ mod \ \lambda(n) < n/2 - \lambda(n)$ with probability $(tp_1q_1 + q_2 + 1/2)/(2tp_1q_1q_2)$, which is negligible. □

Next, we analyze the leakage of a sequence of signatures, which we want to show to be competitive, namely that there are trivial channels in the usage of signatures that allow the leakage of a public key. Next we present a leakage design, based on ordering of signatures only (and not assuming further timing, errors or error correcting based leakage). This shows that in on-going usage there are plenty of opportunities to leak information (which is expected).

Universal attack: this attack works as follows. The user encodes the shadow public key 15 bits at a time, by successively taking 8 unique signatures at a time, and ordering them in a specific way in the post to the bulletin board, say (assuming that scheduling the signatures has this degree of freedom). There are 8! ways to order the signatures (which are distinguished and ordered by their numeric value), so 15 bits can be encoded. This allows us to encode efficiently. For example in 168 signatures we can encode a 314 bit shadow public key (e.g. based on an elliptic curve over GF(2^{157})). This is a "non-cryptographic leakage" which is trivial (and with additional covert information and better encoding we can do with much fewer signatures).

What leakage can we produce in our system? We can, for example leak $\lambda(n)$ over the course of using the signature system, exploiting the structure of our system. In fact, it works as following: When signing a message m_i we produce signature c_i. the attacker gets information (over the integers) of the form $(c_i)^e = x_1h_1(m)+...+x_Kh_K(m)+r_i\lambda(n)$ where the x_i's are fixed and unknown and the unknown last (free) term $(r_i\lambda(n))$ a multiple of $\lambda(n)$ (r_i is associated with c_i). The fixed unknown x_i's can be eliminated from the expressions as additional signatures are added, after $K + 1$ equations, we are left with a large integer representing a multiple of $\lambda(n)$, which can then be used as a public key (and if the process is continued we can get smaller multiples via gcd's and get to isolate $\lambda(n)$). This type of leakage was pointed first by Bleichenbacher. Notice that by choosing K large enough and choosing e large enough we force the attacker to collect a large enough number of signatures and to work over large integers, thus leakage in the scheme more competitive.

There is an obvious tradeoff between efficiency and the competitive factor of this specific leakage scheme. Small K gives a signature whose signing is as expensive as RSA (of double the size) and whose verification takes $e + K$ exponentiations (we implemented the scheme for small values). Finally, we remark that naturally, other leakages may still exist and their presence is an open problem. It is very hard to quantify leakage of this form, and it may be that moderate complexity measures are enough since we deal with prevention in light of existing trivial channels (achieving "competitive leakage").

8 Conclusion

In reaction to NIST's problem, we have presented a framework and an implementation for a digital signature PKI where the public verification keys cannot be abused safely for encryption. We proved its security, argued its no-abuse property, and further argued concretely, based on the sate of the art, why the leakage with on-going use is competitive (compared with universally available means of leakage, which we also pointed out). Numerous issues remain to be investigated like the validation of our no-abuse assumption, whether our assumptions are necessary, and whether or not weaker assumptions can be used. We employed the random oracle proof methodology numerous times regardless of its possible weaknesses. A question one might ask is whether or not there are solutions which do not employ random oracles, yet are efficient, etc. (there may exist a situation where a random oracle is used for arguing that some elements are chosen almost at random, but where standard complexity-theoretic proofs are used for the security argument). The notion of "non-leakage" (or reduced leakage) and its reduction and implementation is another unexplored area: formalization and characterization of the issues, notions, and solutions in the area are left open.

References

[FIPS97] Announcing Plans to Revise Federal Information Processing Standard 186, Digital Signature Standard. In volume 62, n. 92 of *Federal Register*, pages 26293–26294, May 13, 1997.

[An97] R. Anderson. The GCHQ Protocol and Its Problems. In *Advances in Cryptology—Eurocrypt'97*, pages 134-148, 1997. Springer-Verlag.

[BFL91] J. Boyar, K. Friedl, C. Lund. Practical zero-knowledge proofs: Giving hints and using Deficiencies. In *Journal of Cryptology*, 4(3), pages 185–206, 1991.

[BR93] M. Bellare, P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. newblock In *1st Conf. Computer & Comm. Security*, ACM, pages 62–73, 1993.

[BD-etal96] Burmester, Desmedt, Itoh, Sakurai, Shizuya, Yung. A progress report on Subliminal-Free Channels. In *Workshop on Information Hiding*, Cambridge U.K., LNCS, pages 157–168, 1996.

[CGH98] R. Canetti, O. Goldreich, S. Halevi. The Random Oracle Methodology, Revisited. In *ACM STOC '98*.

[C00] J.-S. Coron. On the Exact Security of Full Domain Hash In *Advances in Cryptology—CRYPTO '00*, pages 229–235, 2000. Springer-Verlag.

[D89] Y. Desmedt. Abuses in Cryptography and How to Fight Them. In *Advances in Cryptology—CRYPTO '89*, pages 375–389, 1990. Springer-Verlag.

[DH76] W. Diffie, M. Hellman. New Directions in Cryptography. In volume IT-22, n. 6 of *IEEE Transactions on Information Theory*, pages 644–654, Nov. 1976.

[DSS91] Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). In volume 56, n. 169 of *Federal Register*, pages 42980–42982, 1991.

[ElG85] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology—CRYPTO '84*, pages 10–18, 1985. Springer-Verlag.

[FY95] Y. Frankel, M. Yung. Escrow Encryption Systems Visited: Attacks, Analysis and Designs. In *Advances in Cryptology—CRYPTO '95*, pages 222–235, 1987. Springer-Verlag.

[GHY89] Z. Galil, S. Haber and M. Yung. Minimum-Knowledge Interactive Proof for Decision Problems. In *SIAM Journal on Computing*, 1988.

[GMRi88] S. Goldwasser, S. Micali and R. Rivest. Digital Signature Scheme Secure against Adaptive Chosen Plaintext Attack. In *SIAM Journal on Computing* 1988.

[JY00] A. Juels and M. Yung. Manuscript.

[KL95] J. Kilian and F.T. Leighton. Fair Cryptosystems Revisited. In *Advances in Cryptology—CRYPTO '95*, pages 208–221, 1995. Springer-Verlag.

[L98] A. Lenstra. Generating RSA Moduli with Predetermined Portion. In *Advances in Cryptology—Asiacrypt '98*, pages 1–10, 1998. Springer-Verlag.

[LS] M. Liskov, R. D. Silverman. A Statistical Limited-Knowledge Proof for Secure RSA Keys. Submitted to the IEEE P1363 Working Group. Available at http://grouper.ieee.org/groups/1363/contrib.htm.

[NR94] K. Nyberg, R. Rueppel. Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. In *Advances in Cryptology—Eurocrypt '94*, pages 182–193, 1994. Springer-Verlag.

[NY89] M. Naor, M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. STOC '89, pages 33–43, 19889. ACM.

[Ok92] T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *Advances in Cryptology—Crypto '92*, pages 31–53, 1993. Springer-Verlag.

[PS00] G. Poupard and J. Stern, Short Proofs of Knowledge of Factoring. In *PKC'2000*, LNCS 1751 Springer Verlag, pages 147–166, 2000.

[Ra78] M. Rabin. A Public-key and Signature Scheme as Secure as Factoring, MIT Tech. Report, 1978.

[R98] R. Rivest, Chaffing and Winnowing: confidentiality without encryption. CryptoBytes 4 (1), 1998, pages 12–17.

[RSA78] R. Rivest, A. Shamir, L. Adleman. A method for obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the ACM*, volume 21, n. 2, pages 120–126, 1978.

[Si85] G. Simmons. The Subliminal Channels and Digital Signatures. In *Advances in Cryptology—Eurocrypt '84*, pages 51–57, Springer-Verlag.

[Si93] G. Simmons. Subliminal Communication is Easy Using the DSA. In *Advances in Cryptology—Eurocrypt '93*, Springer-Verlag.

[St96] M. Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology—Eurocrypt '96*, pages 190–199.

[CaSt97] J. Camenisch, M. Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology—Crypto '97*, pages 410–424.

[YY96] A. Young, M. Yung. The Dark Side of 'Black-Box' Cryptography, or: Should We Trust Capstone? In *Advances in Cryptology—CRYPTO '96*,pages 89–103, Springer-Verlag.

[YY97] A. Young, M. Yung. Kleptography: Using Cryptography Against Cryptography. In *Advances in Cryptology—Eurocrypt '97*, pages 62–74, Springer-Verlag.

[YY98] A. Young, M. Yung. Auto-Recoverable and Auto-Certifiable Cryptosystems In *Advances in Cryptology—Eurocrypt '98*, Springer-Verlag.