'LCS/TM-501

LABORATORY FOR COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MIT/LCS/TM-501

A SECURE AND EFFICIENT DIGITAL SIGNATURE ALGORITHM

Silvio Micali

March 1994

READING ROOM
M.I.T.
LABORATORY FOR COMPUTER SCIENCE
545 TECHNOLOGY SQUARE
CAMBRIDGE, MA 02139

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

READING ROOM
M.I.T.
LABORATORY FOR COMPUTER SCIENCE
545 TECHNOLOGY SQUARE
CAMBRIDGE, MA 02139

A Secure and Efficient Digital Signature Algorithm

by

Silvio Micali

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

(Extended Abstract)

March 7, 1994

1 Introduction

This paper puts forward a very efficient and secure digital signature algorithm based both on factoring and hash functions.

Relying on both number theory and hash functions is not, *per se*, a novelty. Besides some important but not-too-practical exceptions (e.g., [1], [6], [10], [13], [14], and [15]), a combination of number theory and one-way hashing is employed by essentially all currently known signature schemes. Some signatures schemes rely on secure hashing only for signing long messages (e.g., [16]). Other schemes make use of hashing only for efficiency purposes (e.g., [11][1]). The new algorithm, instead, *must* hash in order to be secure. This too is not a novel feature, since it is shared by a variety of prior schemes (e.g., [2], [3], [8], [9], and [12] [2]). The real novelty of our scheme lies in the way in which it combines hashing and number theory, so as to optimize, *simultaneously,* many resources: security, efficiency, key generation, and key length.

MAIN FEATURES OF THE NEW ALGORITHM. The new scheme satisfies a very strong security requirement. Namely, if the underlying hash function is secure (e.g., if it behaves like a random function —though a much milder requirement is sufficient) forging signatures is *provably equivalent to factoring.*

On the other hand, the new scheme is very efficient. Key generation essentially consists of selecting two large primes and of performing two modular exponentiations.

[1]This scheme signs every message in two *separate* steps: an off-line step solely based on number theory, and an on-line step solely based on secure hashing.

[2]For instance, if messages are not securely hashed prior to be signed, an enemy can use legitimate signature for forging the signatures of additional messages. Even more dramatically, if message are not hashed prior to be signed, Rabin's scheme is totally breakable by a chosen-message attack.

A public key consists of the product of these two primes, and we recommend this product to be at least 1024-bit long —though it could be shorter in many applications. To achieve a "2^{80} security-level" (assuming that the public key has been chosen long enough not to disrupt this goal, but otherwise *independent* of public-key length), in a typical implementation of the new algorithm a signature needs to be 80-bit longer than the public key; is computed by means of 120 modular multiplications, 80 of which could be performed off-line; and is verified by means of 80 multiplications.

In sum, the new scheme succeeds in being as efficient as the Fiat-Shamir one [8], but is much preferable to it in terms of key length, key generation, and signature length.

ADDITIONAL FEATURES. Our signature scheme has particularly benefited from the algorithmic ideas of Rabin [3], Williams [4], Goldwasser, Micali, and Rivest [6], Goldreich [7], and Fiat and Shamir [8]. Though it makes non-trivial use of number theory, it is *algorithmically very simple*, and can be unexpensively realized in hardware. (Particularly because, being very efficient, it can be implemented by "slower" chips, and because, requiring short secret keys, only a small portion of these chips needs to be "tamper-proof.")

As an extra bonus, the new scheme also is easily transformable into an efficient and secure *identity-based signature scheme* or *identification scheme*. (For reasons of brevity, however, we refrain from deriving the identification scheme explicitly in this extended abstract.)

2 The New Digital Signature Scheme

2.1 Preliminaries

In our signature scheme, we shall make use of some properties and definitions due to Goldwasser, Micali, and Rivest [6], who used them for quite a different scheme.

Definition. Let n be the product of two large primes, $p_1 \equiv 3 \bmod 8$ and $p_2 \equiv 7 \bmod 8$. Define $F_0(x) = x^2 \bmod n$, $F_1(x) = 4x^2 \bmod n$, and, for any non-empty string of bits $\sigma = b_1 b_2 \cdots b_k$, let $F_\sigma = F_{b_k}(\cdots(F_{b_2}(F_{b_1}(x))\cdots)$.

Definition: If n is like above, and x is a square mod n, then exactly one of its four square roots mod n is itself a square mod n, and this root will be denoted by $x^{2^{-1}}$. In general, if k is a positive integer, $x^{2^{-k}}$ will denote the unique square mod n, z, such that $z^{2^k} \equiv x \bmod n$.

The following properties of the above defined quantities have been proved to hold in [6]. However, because their statement and notation in that paper is slightly different from ours, we provide a brief proof of these properties in our appendix.

$\mathcal{A}$: F_0 and F_1 are permutations over the squares mod n, and so is F_σ for any string σ.

$\mathcal{B}$: 2 is not a square mod n, and thus $4^{-2} \not\equiv 2 \bmod n$.

$\mathcal{C}$: If one computes any two squares mod n, x and y, and any two different strings of equal length, σ and τ, such that $F_\sigma(x) = F_\tau(y)$, then one can easily factor n.

The following property $\mathcal{D}$ is due to Goldreich [7]. A brief proof of Goldreich's theorem is also presented in our Appendix.

$\mathcal{D}$: Let k be a positive integer, let $s = 4^{2^{-k}}$ mod n, let σ be a k-bit string, let $int(\sigma)$ be the natural number whose binary representation is σ without its leading 0s (i.e., $int(011) = 3$), and let X and z be squares mod n such that $X = F_\sigma(z)$. Then, $z = X^{2^{-k}}/s^{int(\sigma)}$ mod n.

2.2 The Basic Signature Algorithm

The new digital signature algorithm makes use of a secure hash function H, which could be common to all users. Below, we assume that H produces 80-bit outputs and that "$a \circ b$" denotes the concatenation of strings a and b.

CHOOSING KEYS. Each user randomly selects a large prime $p_1 \equiv 3 \bmod 8$ and a large prime $p_2 \equiv 7 \bmod 8$, and lets $n = p_1 p_2$ be her public key, and $s = 1/4^{2^{-81}} \bmod n$ be her secret key.

> *Comment:* from now on, all computations are mod n unless otherwise specified. The value $4^{2^{-81}}$ can be obtained in three easy steps by (1) computing u_i, the inverse of 2 mod $p_i - 1$, (2) computing $v_i = u_i^{81} \bmod p_i - 1$, and (3) applying the Chinese reminder Theorem to v_1 and v_2 with moduli p_1 and p_2.

SIGNING. A user whose public and secret keys are n and s, respectively, signs a message M as follows.

1. Off-line step. Randomly select a square x and compute $X = x^{2^{81}}$.

   Comment: It is crucial that x, and thus X, be selected afresh for each message to be signed. By construction, $x = X^{2^{-81}}$.

2. On-line step. Compute $\sigma = H(X \circ M)$, $t = s^{int(0\sigma)}$, and $z = xt$, and output (z, σ) as your signature of M.

   Comment: $X \overset{by\,\mathcal{D}}{=} F_{0\sigma}(z) \overset{def.of\,F}{=} F_\sigma(z^2)$. (The "dammy 0" has been added to preserve property $\mathcal{D}$, while avoiding any verification that $F_\sigma^{-1}(X)$ is a square, which is hard without n's factorization.)

VERIFYING. Given a message M and an alleged signature (z, σ) relative to a public key n, compute $X = F_\sigma(z^2)$, and check whether $H(X \circ M) = \sigma$. If so, accept the signature.

2.3 Analysis of the Basic Signature Algorithm

EFFICIENCY. We disregard the single hashing required for signing and verifying, which is trivial. We count instead the number of modular multiplications required by the straightforward implementation of the new algorithm. Notice that the complexity of on-line signing can actually be decreased at will by computing once and for all, and then storing, some suitable values. (As we shall explain in the final paper, the basic algorithm can be modified so that signing can be alternatively/further sped up by increasing signature length.)

- **Off-line signing:** 82 multiplications.

  On-line signing: $\approx$ 121 multiplications (if only the secret key s is stored), or
  $\approx$ 41 multiplications (if the 80 values s^{-2^i} are stored), or
  $\approx$ 11 multiplications (if 2600 convenient values are stored), or ...

- **Verifying:** 81 multiplications.

All multiplications reported above are modulo n. For signing, however, it is more convenient to use the Chinese Reminder Theorem so as to double the small number of required multiplications, but carry them out modulo n's two primes, p_1 and p_2, which are half as long.

SECURITY. The security of the new scheme follows from the "claw-freeness"of the hash function H and the permutations F_0 and F_1. Such a proof of security will be presented in the final paper. Here, let us prove a simpler statement; namely, if H is a random oracle (rather than a function with short description), then the new scheme is *provably* as secure as factoring.[3] Roughly said, what we want to prove is the following: if an enemy has a probability ε of forging a single signature in time T, then he can factor n in expected time $Poly(T/\varepsilon)$.

The key observation to prove this informal statement of security is the following. Assume that an enemy chooses a number z and a 80-bit string σ and computes $X = F_\sigma(z^2)$. Then he will be able to easily sign a message M only if oracle H returns exactly the bit-pattern σ on input $X \circ M$. But this will happen only with probability 2^{-80}, and the adversary cannot easily increase this negligible probability. In fact, if he could concoct a X such that he could forge the signature of any message that, hashed with X, yields either of two different 80-bit patterns, σ and τ, then he should

[3]Indeed, it is a common experience that, if a signature scheme is secure when using a random oracle H, then it continues to be secure when H is approximated by a properly defined one-way hash function. Indeed, this is what it is commonly assumed for schemes like the Fiat-Shamir, the RSA, etc.

be able to produce two pairs (z, σ) and (w, τ) for which $F_\sigma(z^2) = F_\tau(w^2) = X$. But then, by property $\mathcal{C}$, he could easily factor n.

Given the enormously many and unpredictable ways by which an enemy can attack a signature scheme, knowing that no attack could bypass the difficulty of integer factorization if the hash function is secure is of fundamental importance. By comparison, no such proof of security exists for the DSA [16], even even if their hash function is assumed to be a random oracle, and if the discrete logarithm problem is assumed to be intractable. It should also be noted that, similarly to the schemes of Fiat-Shamir [8] and Schnorr's [12], *the new algorithm safely relies on short hashed values.* (For a discussion of this important point, see [12], page 244, or this footnote.[4])

KEYS AND SIGNATURE LENGTH. In the light of the above proof of security, and on the basis of our current knowledge about the factoring problem, in most applications the public key n could be chosen to be 1024-bit long. The corresponding secret key would be equally long, and a signature 1104-bit long.

2.4 Comparison with Other Factoring-Based Schemes

The new scheme is much preferable to the previously-known algorithms based on factoring. For instance, the new scheme is preferable to the RSA from a security point of view. In fact, even if implemented with a random oracle as a hash function, forging messages in the RSA scheme is not known to be equivalent to factoring and it may be much simpler than factoring.

The new scheme is also much preferable to the RSA, Rabin's and William's schemes from the point of view of computational efficiency. In fact, all these schemes sign a message by means of an exponentiation modulo the public key, and no portion of this computation can be performed off-line. Thus, if public keys are chosen 1024-bit long, after hashing the message to be signed, these schemes require, on the average, 1500 modular multiplications for on-line signing. It can then be seen that the new algorithm is 12 times faster than the RSA, Rabin's and William's schemes, even if it does not precompute and store *any* of the s^{2^i} values. If all 81 such values are precomputed, then the new algorithm is 37 times faster than those schemes.

[4]To the contrary, DSA needs to hash messages to at least 160-bit values to be safe against 2^{80}-step attacks against their hash function. In fact, if an enemy keeps on hashing new messages until he finds two messages, M_1 and M_2, such that $H(M_1) = H(M_2)$, then, by asking the signer to sign M_1, he automatically succeeds in forging the signature of M_2. By contrast, in the new scheme —like in the ones of Fiat-Shamir [8] and Schnorr [12]— if an enemy finds two new messages that hash to the same value, he has nothing to gain. In fact, when asked to sign M_1, the legitimate signer will first hash it together with a new random square X; and $H(M_1)$ being equal to $H(M_2)$ does not imply anything about $H(X \circ M_1)$ being equal to $H(X \circ M_2)$. (Similarly, for forging the messages of a new message M, the enemy cannot utilize previous messages M_i that the legitimate signer has signed via their relative squares X_i and hashed values $\sigma_i = H(X_i \circ M_i)$. In fact, if he hashes M with one of the squares used by the legitimate signer —say, X_j— he can successfully exploit the corresponding legitimate signature only with probability 2^{-80}; that is, only if $H(X_j \circ M) = \sigma_j$.)

The new scheme is also preferable to that of Even, Goldreich and Micali [11]. In fact, whether or not its off-line step is implemented with a factoring-based scheme, the latter algorithm produces much longer signatures.

The new scheme also compares very favorably, from an efficiency point of view, with the scheme of Goldwasser, Micali, and Rivest [6]. Indeed, even when this latter algorithm is implemented with Golderich's improvements, it requires more than a modular exponentiation to sign a message, and its signatures are orders of magnitude longer than those of the new scheme.

The new scheme performs as efficiently as the Fiat-Shamir [8] and the Micali-Shamir [9] schemes, but is preferable to both of them in terms of key length and key computation. For instance, if these latter schemes are implemented with $k = 80$ and $t = 1$, then their key generation would require to compute not only two large primes, but also 80 modular exponentiations; in addition, they would require a secret signing key that is at least 40,000-bit long. This can be very cumbersome in some applications —like in smart-card applications, where such a long key needs also to be stored in a tamper-proof storage area.[5]

3 The New Identity-Based Scheme

The new algorithm also yields a new identity-based signature scheme which is more secure and efficient than the original proposal of Shamir [5]. In an identity-based signature scheme[6] there is a special authority, A, and a group of users, $A_1, A_2, \ldots$ Authority A generates a special public-secret key pair (PK, SK), where PK is assumed to be universally known within the system, and then assigns to each user A_i a secret key SK_i, which he can easily compute via SK. Keys PK, SK, and the SK_i's satisfy the following three properties (essentially stating that each user A_i can easily sign, and can be kept accountable by A for what he signs):

1. To each user A_i corresponds a key PK_i that is easily computable on inputs i and PK.

2. User A_i, thanks to his special knowledge of SK_i, can easily sign, relative to PK_i, any message m.

3. User A_i cannot forge any signature (relative to PK_j) of another user A_j.

[5]Of course, the Fiat-Shamir and the Micali-Shamir schemes could achieve a better performance in both key generation and key length by choosing smaller values of k and correspondigly larger values for t; but, in this case, their signatures would become *considerably longer* than in the new scheme. In any case, the new algorithm, similarly modified, would match the computational efficiency and signature length of these algorithms, while still beating them in key length and key generation.

[6]Note that Shamir never precisely defined his notion; moreover, he focused primarily on public-key cryptosystems rather than on digital signature schemes. It should thus be expected that our (still-informal) description of an identity-based signature scheme may express more our own desiderata than what Shamir originally intended.

To these basic properties we wish to add the following two optional ones.

4. The users cannot use the keys that allow them to sign in order to encrypt messages to each other.

5. A can be conveniently replaced by a group of "trustees."

If Property 5 is not properly enforced, then authority A can also forge signatures for its users, and thus the applicability of identity-based schemes is primarily confined to those systems which have a well-defined and well-trusted authority. For instance, when A is a bank and the A_i's its branches, or when A is the Government and the A_i Government agencies or employees.

We shall show how property 5 can be easily enforced in the final paper. For the time being, let us see how the other properties can be achieved.

3.1 The Basic Identity-Based Algorithm

While Shamir exemplified identity-based signature schemes based on the RSA function and some additional and little-understood assumption, we will succeed in implementing them in a way that is provably equivalent to factoring, at least when one hashes by means of a random oracle H.

We start by pointing out that, in the scheme of Section 2, one can replace the number 4 with a random square mod n, S, without loosing security. In fact, if a -1 Jacobi symbol root of S, r, is publicly known (in the basic new scheme, 2 is guaranteed to be a square root of 4 with Jacobi symbol -1), then the same proof of security of the basic new scheme applies. (In essence, if one finds squares x and y such that $x^2 = Sy^2$, then one finds a Jacobi-symbol $+1$ root of S, since $x/y = S^{2^{-1}}$. Thus, $gcd(x/y+r, n)$ is a proper factor of n.) On the other side, if S is chosen so that no enemy knows any square root of it mod n, then, for what we just said, finding two squares x and y such that $x^2 = Sy^2$ is tantamount to finding a square root of S. But, as shown by Rabin [3], finding square roots of an arbitrary square mod n is as hard as factoring n.

In order to replace 4 with a random square S, we shall use the following property due to [4].

$\mathcal{E}$: Let n be the products of two primes, one congruent to 3 mod 8 and the other to 7 mod 8. Then, for any member of Z_n^*, exactly one of $x, -x, 2x, -2x$ mod n is a square mod n.

Assume now that A has selected one such n, and has stored its secret prime factorization (which needs not to be done in the basic new scheme). Then, A allows his users to share essentially his same public key n as follows.

ASSIGNING KEYS. A computes $S_i = H(n \circ i)$ (i.e., the hash of the concatenation of n and the name of the user himself). Then, A computes $\alpha_i \in \{1, -1, 2, -2\}$ such that $\alpha_i S_i$ is a square mod n, computes the value $s_i = 1/(\alpha_i S_i)^{2^{-81}}$, and gives s_i to A_i as a secret key. The corresponding public key of A_i will be (n, i, α_i).

(*Comment:* if n is universally known, the recipient of a signature of A_i knows all of A_i's public key, except for the two bits needed to specify α_i.) It is not crucial that $S_i = H(n \circ i)$, but it is important that S_i be "sufficiently random, though easily computable on input i," so that i's secret key is sufficiently unpredictable and unrelated to that of j.

SIGNING. A_i signs a message M as in the basic new scheme, but using s_i instead of $4^{2^{-81}}$ and S_i instead of 4. To the resulting signature, (z, σ), A_i also adds its own α_i.

VERIFYING. To verify the signature $((z, \sigma), \alpha_i)$ of A_i for message M, the verifier checks that $\alpha_i \in \{1, -1, 2, -2\}$, computes $S_i = H(n \circ i)$ and $\alpha_i S_i$. Then, he computes $X = F_\sigma(z^2)$, but using $F_0(x) = x^2$ and $F_1(x) = \alpha_i S_i x^2$ as basic permutations, and accepts $((z, \sigma), \alpha_i)$ to be A_i's signature of M if $H(X \circ M) = \sigma$.

Notice that signing will require exactly the same amount of multiplications as in the basic new scheme; that is, 82 multiplications off-line and either 121 or 40, depending on storage, on-line. Verifying will instead require 121 multiplications rather than 81. (Indeed, on the average, the function F_1 will need to be applied 40 times, but each application, rather than costing one multiplication —as when $F_1(x)$ were $4x^2$,— now costs two multiplications, since now $F_1(x) = Sx^2$.)

It is now easy to see that Properties 1—3 are satisfied. The main idea behind proving that Property 4 also holds is simple:

All users in the group share the same public key n without knowing its factorization.

Details will be given in the final paper.

3.2 Applications to Public-Key Certification

Identity-based signature schemes like the one above also enjoy the following important property:

There is no need to certify individual public keys for the users in the group.

That is, only n needs to be certified outside the group, if it is not already universally known. This property makes such algorithms very suitable for public-key certification. Indeed, in a digital signature scheme, the public verification-keys of the users need to be certified, since it is unlikely that they are universally known. For instance, a user U may have his public key PK_U signed by a local authority A_i, relative to this authority's public key PK_i. However, since there may be thousands of local authorities, their public keys may not be universally known either, and thus A_i's signature does not quite certify PK_U. It maybe assumed, however, that the public key of central authority A is universally knwon; thus A may easily sign the local authorities public keys, and a certificate of PK_U may consist of (1) A_i's signature of PK_U, (2) PK_i and (3) A's signature of PK_i. Thus, such a certificate of PK_U is rather long. This is unfortunate, since most of the time user U sends a signature of his, he must also send this certificate along; and this certificate will need to be stored by the recipient for a long time. Thus these long certificates are very costly, both in terms of transmission and in terms of storage. An identity-based scheme like the one above, may instead reduce these costs by two thirds. For instance, the local authorities (e.g., the post offices) may use the new identity-based scheme for signing messages, and since the composite number n chosen by the central authority is universally known, the certificate of PK_U may just be i and A_i's signature of PK_U.

Notice that string i is very short (in the case of a post office may coicide with its ZIP code). Moreover, the recipient of the certificate needs not to verify whether a post office with that ZIP code exists: if central authority A had computed the right secret key for "i" this means that "i" was a legitimate name. Finally, notice that in this application there is a natural and trusted central authority, and thus the issue of replacing A with a group of trustees may not be relevant.

Indeed, the new algorithm offers such a simple key certification (and the new circuitry sufficiently unexpensive) that it is almost worth using it even if another signature scheme were adopted for the individual users.

Acknowledgements. Special thanks to Mihir Bellare and Oded Goldreich. Thanks also to Shafi Goldwasser and Philip Rogaway.

References

[1] L. Lamport. *Constructing Digital Signatures from a One-Way Function.* SRI Intl. CSL-98.

[2] R. Rivest, A. Shamir, and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Comm. ACM, Vol. 21, 1978, pp. 120-126.

[3] M. Rabin. *Digitalized Signatures as Intractable as Factorization.* MIT Laboratory for Computer Science Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge, MA, January 1979.

[4] H.C. Williams. *A modification of the RSA Public-Key Cryptosystem.* IEEE Trans. Inform. Theory, IT-26, 1980, pp. 726-729.

[5] A. Shamir. *Identity-Based Cryptosystems and Signature Schemes.* Proc. Crypto 84.

[6] S. Goldwasser, S. Micali, and R. Rivest. *A Digital Signature Secure Against Adaptive Chosen-Message Attacks.* SIAM J. Comput., Vol. 17, No. 2, April 1988, pp. 281-308.

(A preliminary version of this paper, *A Paradoxical Solution to the Sigature Problem*, appeared in the Proc. of the 25th Annual Symp. on the Foundations of Computer Science, November 1984.)

[7] O. Goldreich. *Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme.* Proc. Crypto 86, pp. 104-110.

[8] A. Fiat and A. Shamir. *How to Prove Yourselves: Practical Solutions of Identification and Signature Problems.* Proc. Crypto 86, Springer Verlag, 263, 1987, pp.186-194.

[9] S. Micali and A. Shamir. *An Improvement of the Fiat-Shamir Identification and Signature Scheme.* Proc. Crypto 88.

[10] M. Bellare and S. Micali. *How to Sign Given any Trapdoor Function.* SIAM J. on Computing, 1992, Vol. , pp.

(A preliminary version of this paper, *How to Sign Given any Trapdoor Function,* appered in the Proc. of 20th Annual Symp. on the Theory of Computing, ACM, 1988, pp. 32-42.)

[11] S. Even, O. Goldreich, and S. Micali. *On-Line/Off-Line Digital Signatures.* Proc. Crypto 89. Springer Verlag, 1990.

[12] C. Schnorr. *Efficient Identification and Signatures for Smart Cards.* Proc. Crypto 89. Springer Verlag, 1990.

[13] R. Merkle. *A Certified Digital Signature.* Proc. Crypto 1989. Springer Verlag, 1990.

[14] M. Naor and M. Yung. *Universal Hash Functions and Their Cryptographic Applications.* Proc. 21st Annual Symp. on The Theory of Computing. ACM, 1989, pp. 33-43.

[15] J. Rompel. *One-Way Functions Are Necessary and Sufficient for Secure Signatures.* Proc. 22nd Annual Symp. on the Theory of Computing, ACM, 1990, pp. 387-394.

[16] National Institute of Standard and Technology. *A Digital Signature Standard.* Well-described in *Debating Encryption Standards*, Comm. ACM, July 1992, Vol. 35, No. 7, pp. 33-54.

4 Appendix

Many of the properties listed in Section 1 can be established based on the the Jacobi-symbol function. Let us recall some well-known facts about it. If p is a prime and x a integer mod p, then the Jacobi symbol of x mod p, denoted here by (x/p), equals $x^{(p-1)/2}$ mod p. Thus, by virtue of Euler's theorem, $(x/p) = +1$ if x is a square mod p, and -1 otherwise. The Jacobi-symbol function can be extented to Z_n^* when n is composite. In our case, since $n = p_1p_2$, this extension is simply described as $(x/n) = (x/p_1)(x/p_2)$. Thus, a necessary condition for x to be a square mod n is that $(x/n) = +1$. (In fact, x is a square mod n if and only if it is a square mod both p_1 and p_2, and if $(x/n) = -1$, then $(x/p_i) = -1$ for either $i = 1$ or $i = 2$.) On the other side, this condition is not sufficient. (In fact, if $(x/p_1) = (x/p_2) = -1$, then $(x/n) = +1$, but x is not a square.) The Jacobi symbol is also multiplicative on its first argument (i.e., $(x_1x_2/n) = (x_1/n)(x_2/n)$). Finally, from these basic facts, one can easily derive the following properties about those integers n which are product of one prime congruent to 3 mod 8 and another prime congruent to 7 mod 8:

(*i*) $(-1/n) = +1$, but -1 is not a square mod n.

(*ii*) If $x, -x, y$, and $-y$ are the four square roots of a given square mod n, then exactly one of them is a square mod n.

(*iii*) $(2/n) = -1$.

Properties $\mathcal{A}$, $\mathcal{B}$,$\mathcal{C}$, and $\mathcal{D}$ are then immediately derivable from the above simple facts.

Proof of $\mathcal{A}$: F_0 is a permutation over the squares of n due to (*ii*). F_1 is a permutation over the squares of n because it is the composition of two permutations over the squares mod n: F_0 and multiplication by 4. (In fact, multiplication by a fixed integer is a permutation over Z_n^*, and since 4 is a square mod n, multiplication by 4 mod n is a permutation over the squares mod n.) For any string σ, F_σ is a permutation over the squares mod n because it is a composition of F_0 and F_1.

Proof of $\mathcal{B}$: Straightforward from (*iii*), the definition of x^{-2}, and the fact that Jacobi-symbol -1 elements cannot be squares mod n.

Proof of $\mathcal{C}$: Let us first rephrase $\mathcal{C}$ a bit more precisely. Namely, if x and y are squares mod n, and σ and τ are two different strings, of equal and positive length, such that $F_\sigma(x) = F_\tau(y)$, then there exists two shorter, equal-length, prefixes (possibly empty) of σ and τ, respectively, σ' and τ', such that $gcd(F_{\sigma'}(x)+2F_{\tau'}(y), n)$ is a proper factor of n —and thus equal to either p_1 or p_2. (Since, for uniformity of presentation, we considering empty prefixes, we need to enlarge our definition of F_σ as follows: if ϕ is the empty string, then we define F_ϕ to be the identity.)

The proof is by induction over the length of σ and τ. We start with proving the base case: $|\sigma| = |\tau| = 1$. Assume that x and y are squares mod n and that

$F_0(x) = F_1(y)$, that is, $x^2 \equiv 4y^2 \bmod n$. Then x and $2y$ are square roots of a common square mod n. Moreover, $x \not\equiv 2y \bmod n$. (In fact, $(x/n) = +1$ because x is a square, but $(2y/n) = (2/n)(y/n) = -1$, because of (iii) and the fact that y is a square.) Similarly, $x \not\equiv -2y \bmod n$. (In fact, $(x/n) = +1$ because x is a square, but $(-2y/n) = (-1/n)(2/n) = -1$, because what we have observed about -1 and $2y$.) Thus, by a well-known theorem, we have, except for renaming, $gcd(F_\phi(x) + 2F_\phi(y), n) = gcd(x + 2y, n) = p_1$ and $gcd(F_\phi(x) - 2F_\phi(y), n) = gcd(x - 2y, n) = p_2$. This proves the base case. Assume now that Property $\mathcal{C}$ holds for $0 < |\sigma| = |\tau| \leq k$, and let us show that it holds for $k+1$. Let σ and τ be two different bit-strings of length $k+1$, let x and y be two squares such that $F_\sigma(x) = F_\tau(y) = z$, and let σ' and τ' be, respectively, the k-bit prefixes of σ and τ. Then, if $F_{\sigma'}(x) = F_{\tau'}(y)$, we are done because of our inductive hypothesis. If, instead, $x' = F_{\sigma'}(x) \neq F_{\tau'}(y) = y'$, then x' and y' are squares mod n (by Property $\mathcal{A}$), and the last bit of σ is different from the last bit of τ. (In fact, if b were the last bit of both strings; then permutation F_b should map the two different squares x' and y' to the same value z, since $F_b(x') = F_b(F_{\sigma'}(x)) = F_\sigma(x) = z = F_\tau(x) = F_b(F_{\tau'}(x) = F_b(y')$.) But then, assuming without loss of generality that $\sigma = \sigma'0$ and $\tau = \tau'1$, the two squares x' and y' are such that $F_0(x') = z = F_1(y')$. Thus by the already proven base case, $gcd(x' + 2y', n)$ is a proper factor of n. In virtue of our definition of x' and y', this proves that $\mathcal{C}$ holds for $k+1$

Proof of $\mathcal{D}$: By induction on k, the length of σ. (All computations are mod n.) For proving the base step, $k = 1$, set $s = 4^{2^{-1}}$ and let σ be a bit. If $\sigma = 0$, then $z = F_0^{-1}(X)$. Thus, because X and z are both squares and because of the definition of the $(\cdot)^{2^{-1}}$ operator, $z = X^{2^{-1}}$. And by re-writing the last equation we obtain

$$z = X^{2^{-1}}/1 = X^{2^{-1}}/s^0 = X^{2^{-1}}/s^{int(0)}.$$

If $\sigma = 1$, then $z = F_1^{-1}(X)$. Thus, because $X/4$ and z are both squares and because of the definition of the $(\cdot)^{2^{-1}}$ operator, $z = (X/4)^{2^{-1}}$. And by re-writing the last equation we obtain

$$z = X^{2^{-1}}/4^{2^{-1}} = X^{2^{-1}}/s = X^{2^{-1}}/s^1 = X^{2^{-1}}/s^{int(1)}.$$

Thus $\mathcal{D}$ holds for $k = 1$. Assume now that $\mathcal{D}$ holds for $k > 0$. For proving that $\mathcal{D}$ holds for $k+1$, set $s = 4^{2^{-(k+1)}}$, and let σ be an arbitrary k-bit string.

If $X = F_{0\sigma}(z)$, then $int(0\sigma) = int(\sigma)$ and $z = (F_\sigma^{-1}(X))^{2^{-1}}$. Thus, by inductive hypothesis, $z = (X^{2^{-k}}/4^{2^{-k}int(\sigma)})^{2^{-1}}$. By re-writing this last equation we obtain

$$z = X^{2^{-(k+1)}}/4^{2^{-(k+1)}int(\sigma)} = X^{2^{-(k+1)}}/s^{int(\sigma)} = X^{2^{-(k+1)}}/s^{int(0\sigma)}.$$

If $X = F_{1\sigma}(z)$, then $int(1\sigma) = 2^k int(\sigma)$ and $z = (F_\sigma^{-1}(X)/4)^{2^{-1}}$. Thus, by inductive hypothesis, $z = ((X^{2^{-k}}/4^{2^{-k}int(\sigma)})/4)^{2^{-1}}$. By re-writing this last equation we obtain

$$z = (X^{2^{-(k+1)}}/4^{2^{-(k+1)}int(\sigma)})/4^{2^{-1}} = X^{2^{-(k+1)}}/s^{int(\sigma)}4^{2^{-1}} =$$
$$X^{2^{-(k+1)}}/s^{int(\sigma)}s^{2^k} = X^{2^{-(k+1)}}/s^{int(\sigma)+2^k} = X^{2^{-(k+1)}}/s^{int(1\sigma)}.$$

This completes the proof of the inductive step.