Appeared in *Advances in Cryptology – Crypto '86 (Proceedings)*, (A.M. Odlyzko ed.), Lecture Note in Computer Science (263) Springer Verlag, pages 104–110, 1987.

Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme

(Reproduced from preprint)

Oded Goldreich

Department of Computer Science and Applied Mathematics
Weizmann Institute of Scienve, Rehovot, Israel

ABSTRACT

The focus of this note is the Goldwasser-Micali-Rivest Signature Scheme (presented in the *25th POCS*, 1984). The GMR scheme has the salient property that, unless factoring is easy, it is infeasible to forge any signature even through an adaptive chosen message attack. We present two technical contributions with respect to the GMR scheme:

1) *The GMR scheme can be made totally "memoryless"*: That is, the signature generated by the signer on message M does not depend on the previous signed messages. (In the original scheme, the signature to a message depends on the number of messages signed before.)
2) *The GMR scheme can be implemented almost as efficiently as the RSA*: The original implementation of the GMR scheme based on factoring, can be speeded-up by a factor of $|N|$. Thus, both signing and verifying take time $O(|N|^3 \log^2 |N|)$. (Here N is the moduli.)

Work done while author was in the Laboratory for Computer Science, MIT.
Partially supported by a Weizmann Postdoctoral Fellowship, an IBM Postdoctoral Fellowship, and NSF Grant DCR-8509905.

1. Introduction

In 1984, Goldwasser, Micali and Rivest presented a signature scheme robust against adaptive chosen message attack [GMR]: *no adversary who first receives signatures to messages of his choice, can later create a signature of even a single additional message.* The scheme uses two pairs of trapdoor permutations, and is proven to be secure (in the above sense) if these pairs have a certain clawfree property (i.e. it is infeasible to find x and y such that $f_0(x) = f_1(y)$). It was shown that the intractability assumption of factoring implies the clawfree condition, and thus a concrete factoring-based implementation was presented.

The original GMR scheme suffers from two aesthetic drawbacks:

1) The signature scheme is not completely "memoryless". That is, the signature generated by the signer slightly depends on the previous signed messages.
2) The signing process in the suggested factoring-based implementation is too slow. Let n be the security parameter (i.e. the length of the moduli), then signing takes more than n^4 steps (as opposed to n^3 steps in the RSA).

In this note we suggest a modification of the abstract GMR scheme, and a speed-up in its factoring-based implementation. These suggestions eliminate the drawbacks listed above, while maintaining the security of the original scheme. That is, the modified scheme is totally "memoryless", and its implementation using factoring is almost as fast as the RSA (while being much more secure than RSA!).

The rest of this note is organized as follows. In Section 2 we briefly review the GMR signature scheme. The reader is encourage to consult the original paper [GMR] for a more complete exposition of the GMR scheme. (This reference is also an excellent source for the critical review and a historical account of the problem of obtaining digital signatures.) In Section 3, we present a modification which makes the GMR scheme "memoryless". In Section 4, we present a speed-up in the factoring-based implementation of the GMR scheme. Our conclusions are presented in Section 5.

2. Overview of the GMR Scheme

The GMR scheme is basically a two-stage authentication process. First the signer's entry in the public file is used to authenticate a random *point of reference* (hereafter called *REF*). Next this REF is used to authenticate the message in a bit-by-bit manner. The same REF is never used to authenticate two different messages. The scheme utilizes two pairs of trapdoor clawfree permutations, denoted (f_0, f_1) and (g_0, g_1). The f-pair is used for the first stage (authentication of the REF), while the g-pair is used for the second stage (authentication of the message).

The REFs are generated from the public file by using a tree structure (hereafter called the *REF tree*). The same REF tree is used to sign all messages. The root of the REF tree contains part of the signer's (entry in the) public file. Each internal node in the REF tree has a constant number of children (say three). Leaves in the REF tree are used to authenticate messages (in the second stage of the signing process). It was originally proposed [GMR] that only the third node of each internal node be a leaf, while the other children be potential internal nodes. As we will see in the sequel, this particular tree structure (for the REF tree) is not essential.

Authentication of the nodes in the REF tree is done successively: each node is authenticated by its father in the tree. A crucial detail in the GMR scheme is the manner in which one REF authenticates its children. Each REF is partitioned into five parts: its "name" (denoted x), the names of its children (denoted y_1, y_2, y_3), and a "tag" (denoted t) binding them all together. The *tag* is computed using the first pair of (trapdoor) clawfree permutations f_0 and f_1, through what is called *an authentication step*.

Let $F^{-1}(\sigma, x) = f_\sigma^{-1}(x)$, for every $\sigma \in \{0,1\}$. Let $F^{-1}(\alpha\sigma, x) = F^{-1}(\alpha, f_\sigma^{-1}(x))$, for every $\sigma \in \{0,1\}$ and $\alpha \in \{0,1\}$.

The the five parts of the REF satisfy $t = F^{-1}(y_1 y_2 y_3,\ x)$.

(The REF tree should not be confused with the tree-like nature of the authentication step.) The authentication of the message, in the second stage, consists of a single authentication step that uses the g-pair. This authentication step binds the message m (to be signed) to a leaf in the REF tree. The tag t binding the leaf named x with the message m satisfies $t = G^{-1}(m, x)$, where G^{-1} is defined analogously to F^{-1} based on the permutations g_0 and g_1.

To sign a new message, m, the signer identifies a leaf that was not used so far (named z). If no such leaf exists, the signer identifies an unused internal node names x (i.e. a node which is a potential internal node but has no children yet), randomly selects names for its children (y_1, y_2, y_3), computes a tag binding the children to their father (i.e. uses the trapdoor information to compute $t = F^{-1}(y_1, y_2, y_3,\ x)$), stores the names of the children, and sets $x = y_3$. The signer retrieves (or recomputes) the tags for all the authentication steps leading from the root of the REF tree to the leaf named z. (It is crucial that, in all signatures produced, the same names are used for the same nodes.) This completes the first stage of the signing process. Next the signer uses the trapdoor information (for the pair g_0 and g_1) to compute the tag $G^{-1}(m, z)$. The signature consists of the list of all authentication steps mentioned above (corresponding to a path from the root to a leaf and an extra step authenticating the message by the leaf).

To verify the validity of a signature, the verifier checks that the list corresponds to a path from the root to a leaf, and that all tags along this path are valid. To validate $t = F^{-1}(\alpha, x)$, the verifier computes $F(\alpha, t)$ and compares it to x.

3. Making the GMR Scheme Memoryless

As hinted in the outline of the GMR scheme, the particular way of using the nodes in the REF tree is not essential to the scheme. In the original scheme, the nodes were used in a "greedy" manner so to minimize the length of the first signatures. The drawback in that suggestion is that it was required to remember the identity of the last node used for message authentication (i.e. to store the number of messages signed so far). Our aim is to eliminate the dependency of the next signature on the past. Thus, we suggest to use the REFs in the tree differently than in the original scheme.

Let n be the security parameter, and let $k = (\log_2 n)^2$. For message authentication (second stage), we will use only the REFs in the kth level of the REF tree. In other words, the REF tree will be a full binary tree of depth k. The key idea is to use a randomly chosen

(k-level) leaf in order to authenticate a new message. With very high probability, we will never use the same leaf twice (in the second stage). This is the case since 2^k was chosen to be much larger than the number of messages to be ever signed.

The last detail to be mentioned is that the names of the nodes in the REF tree should be generated using a pseudorandom function (applied to the location of the node), rather than randomly. This way, we guarantee that the same names are always used for the same nodes, without having to store these names. (The idea to use pseudorandom functions to eliminate this part of the storage requirement in the GMR scheme was suggested by Leonid Levin.) Assuming the existence of one-way permutations, pseudorandom functions can be efficiently constructed [GGM].

Proving that the modified signature scheme is as secure as the original scheme requires a two-step argument. First, one should consider a hybrid scheme where the names of the nodes in the REF tree are generated randomly as in the original scheme. The proof of security for this scheme is conceptionally identical to the proof presented in [GMR], and is only a little more involved in details. Next, one uses the polynomially indistinguishability of the pseudorandom functions (from random functions), to show that the proposed scheme is as secure as the hybrid scheme.

Remark: the identity of the (k-level) leaf to be used can be computed by applying a pseudorandom function to the message to be signed. Thus, no coin tosses are required during the signing process. One can easily show that the security feature is preserved.

4. Speeding-up the Signing Process in the Factoring Based Implementation

The computational bottleneck in the signing/verifying process are, respectively, the computation of $F^{-1}(\cdot,\cdot)$ given the trapdoor and the computation of $F(\cdot,\cdot)$. In the factoring-based implementation presented in [GMR], $F^{-1}(\cdot,\cdot)$ was computed in $|N|^4$ steps, while $F(\cdot,\cdot)$ was computed in $|N|^3$ steps, where N is the modulus in use. In this section we show how to compute F^{-1} in $|N|^3$ steps.

In the implementation based on factoring, $N = pq$ is the product of two primes satisfying $p \equiv q \equiv 3 \ (mod\ 4)$ and $p \neq q \ (mod\ 8)$. This way, each quadratic residue mod N has a unique square root which is a quadratic residue itself, and neither $+2$ nor -2 is a quadratic residue mod N. f_0 and f_1 are defined to be permutations over the set of quadratic residues mod N:

$$f_0(x) = x^2 \ (mod\ N)$$

and

$$f_1(x) = 4 \cdot x^2 \ (mod\ N)\ .$$

Let $\sqrt{x}$ denote the square root of x which is a quadratic residue mod N. Thus, $f_0^{-1}(x) = \sqrt{x}$ and $f_1^{-1}(x) = \sqrt{x/4}$. Note that $\sqrt{4} \neq 2 \ (mod\ N)$. This observation is the key for proving that, unless factoring is easy, the permutations f_0 and f_1 defined above constitute a pair of (trapdoor) clawfree permutations. For more details see [GMR].
Recall the definition of $F^{-1}(\alpha, x)$

$$F^{-1}(\sigma_1\sigma_2\cdots\sigma_n,\ x) = f_{\sigma_1}^{-1}\left(f_{\sigma_2}^{-1}(\cdots f_{\sigma_n}^{-1}(x)\cdots)\right)\ .$$

The obvious way to compute $F^{-1}(\alpha, x)$ is by successively taking square roots. This results in $\Theta(|\alpha|)$ exponentiations. We present an alternative and faster way for computing $F^{-1}(\alpha, x)$. This way requires only a constant number of exponentiations, and is based on the following observation:

Let M denote the length of α, and $i(\alpha)$ denote the integer naturally encoded by the string α. Let $R_N(2^m, z)$ denote the 2^mth root of z modulo N (i.e. the quadratic residue obtained from z by repeating the $\sqrt{\cdot}$ operator m times). Then

$$F^{-1}(\alpha, x) = \frac{R_N(2^m, x)}{(R_N(2^m, 4))^{i(\alpha)}} \ (mod\ N) \ .$$

(See example in the Appendix.)

Computing $R_N(2^m, z)$ reduces to computing it modulo each of the factors (i.e. computing $R_p(2^m, z)$ and $R_q(2^m, z)$ and applying the Chinese Reminder Theorem. Rather than computing $R_p(2^m, z)$ by successive applications of $\sqrt{\cdot}$, we compute it by one exponentiation:

(Pre)-compute, $inv_p(2) = (p+1)/4$, the "inverse" of 2 modulo $\phi(p) = p - 1$).

(Note that $R_p(2, z) = z^{inv_p(2)}$.)

(Pre)-compute, $inv_p(2^m) = (inv_p(2))^m \ mod\ p - 1$, the "inverse" of 2^m modulo $\phi(p)$.

Compute $r_p = z^{inv_p(2^m)} \ (mod\ p)$.

(Clearly, $r_p = R_p(2^m, z)$.)

Thus, computing F^{-1} reduces to a constant number of modular exponentiations.

5. Conclusions

Incorporating the two modifications into the GMR signature scheme, we get a scheme which is as secure as the original one, and is both "memoryless" and "practical". (It is important to note that assuming the intractability of factoring, pseudorandom functions f can be implemented, and that evaluating $f(\beta)$ can be done in $n^3 \cdot |\beta|$ steps. Also note that the length of the argument to f is $k = (\log_2 n)^2$.)

The "dependency on the past" in the original GMR scheme has nothing to do with the resolution of the "Paradox" mentioned in [GMR]. The paradox is resolved by observing that the adversary is uniform, while the real signer is "non-uniform". For further discussion see [GMR].

Throughout this note, we have implicitly assumed that the length of the message to be signed is linear in the security parameter (n). However, the GMR scheme works also if this is not the case, while the running time (naturally) increases. In case we are using the factoring-based implementation and the message has length $m \gg n$, the signing time increases by an additive term of $O(m \cdot n)$ and the verification time increases by an additive term of $O(m \cdot n^2)$. A different approach suggested recently by Damgard is to first hash the message using *collision free hash functions* and then to sign the hashed value [D]. Interestingly, Damgard also shows that such hash functions can be constructed based on the existence of any pair of clawfree permutations.

ACKNOWLEDGEMENTS

Section 4 is joint work with Shafi Goldwasser. I would like to thank her for the collaboration, and for the permission to present the result here. I would also like to thank Louis Guillou for suggesting a simplification in the presentation of this result.

I am grateful to Shafi Goldwasser, Silvio Micali and Ron Rivest for many illuminating discussions concerning their signature scheme.

REFERENCES

[D] Damgard, I.B., "Collision Free Hash Functions and Public Key Signature Schemes", manuscript, 1986.

[DH] Diffie, W. and M.E. Hellman, "New Directions in Cryptography", *IEEE Trans. Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644–654.

[GGM] Goldreich, O., S. Goldwasser and S. Micali, "How to Construct Random Functions", *Proc.* 25^{th} *Symp. on Foundation of Computer Science*, 1984, pp. 464–479. To appear in *J. ACM*.

[GMR] Goldwasser, S., S. Micali and R.L. Rivest, "A *Paradoxical* Solution to the Signature Problem", *Proc.* 25^{th} *Symp. on Foundation of Computer Science*, 1984, pp. 441–448. A better version is available from the authors.

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

APPENDIX

Recall the definition of $F^{-1}(\alpha, x)$

$$F^{-1}(\sigma_1\sigma_2\cdots\sigma_n,\ x) = f_{\sigma_1}^{-1}\Big(f_{\sigma_2}^{-1}\big(\cdots f_{\sigma_n}^{-1}(x)\cdots\big)\Big)$$

and

$$f_0^{-1}(x) = \sqrt{x} \qquad \text{and} \qquad f_1^{-1}(x) = \sqrt{x/4}\ ,$$

where $\sqrt{z}$ is the square root of z which is a quadratic residue modulo N. Let us consider the case where the length of α is 2. we get,

$$F^{-1}(00, x) = f_0^{-1}\big(f_0^{-1}(x)\big) = \sqrt{\sqrt{x}} = \frac{R_N(2^2, x)}{(R_N(2^2, 4))^0}\ (mod\ N)$$

$$F^{-1}(01, x) = f_0^{-1}\big(f_1^{-1}(x)\big) = \sqrt{\sqrt{x/4}} = \frac{R_N(2^2, x)}{(R_N(2^2, 4))^1}\ (mod\ N)$$

$$F^{-1}(10, x) = f_1^{-1}\big(f_0^{-1}(x)\big) = \sqrt{\sqrt{x}/4} = \frac{R_N(2^2, x)}{(R_N(2^2, 4))^2}\ (mod\ N)$$

$$F^{-1}(11, x) = f_1^{-1}\big(f_1^{-1}(x)\big) = \sqrt{\sqrt{x/4}/4} = \frac{R_N(2^2, x)}{(R_N(2^2, 4))^3}\ (mod\ N)\ .$$