Fast Signature Generation with a Fiat Shamir – Like Scheme

H. Ong
Deutsche Bank AG
Stuttgarter Str. 16–24
D – 6236 Eschborn

C.P. Schnorr *
Fachbereich Mathematik / Informatik
Universität Frankfurt
Postfach 111932
D – 6000 Frankfurt/M. 11

Abstract

We propose two improvements to the Fiat Shamir authentication and signature scheme. We reduce the communication of the Fiat Shamir authentication scheme to a single round while preserving the efficiency of the scheme. This also reduces the length of Fiat Shamir signatures. Using secret keys consisting of small integers we reduce the time for signature generation by a factor 3 to 4. We propose a variation of our scheme using class groups that may be secure even if factoring large integers becomes easy.

1 Introduction and Summary

The Fiat–Shamir signature scheme (1986) and the GQ–scheme by Guillou and Quisquater (1988) are designed to reduce the number of modular multiplications that are necessary for generating signatures in the RSA–scheme. Using multicomponent private and public keys Fiat and Shamir generate signatures much faster than with the RSA–scheme. The drawback is that signatures are rather long. They are about t–times longer than RSA–signatures, where t is the round number in the Fiat–Shamir scheme. Using single component keys Guillou and Quisquater obtain signatures of about the same length as in the RSA–scheme but the cost for signature generation is only slightly reduced (by a factor of about 3) compared to the RSA–scheme.

In this paper we propose a new signature scheme and a corresponding authentication scheme that reduces the length of signatures in the Fiat–Shamir scheme to about the length of RSA–signatures. Signature generation with the new scheme is about

*This research was performed while the second author visited the Department of Computer Science of the University of Chicago

I.B. Damgard (Ed.): Advances in Cryptology - EUROCRYPT '90, LNCS 473, pp. 432-440, 1991.

3 to 4 times faster than with the Fiat–Shamir scheme. The efficiency of the new signature scheme is comparable to that of the discrete logarithm signature scheme by Schnorr (1989): In the new scheme signature generation is somewhat slower, signature verification about 5 times faster than in the discrete logarithm scheme. Signatures, private and public keys are longer in the new scheme.

We present the basic version of the new signature scheme in section 2. This basic version preserves the efficiency of the Fiat–Shamir scheme but reduces the length of signatures. In section 3 we present a variant of the new scheme that generates signatures about 3 to 4 times faster than with the Fiat–Shamir scheme. The authentication scheme that corresponds to the signature scheme is presented in section 4. It is shown to be secure unless computing non trivial 2^t-th roots modulo N is easy. A variation of our scheme using class groups is given in section 5. This variant may be secure even if factoring large integers is easy.

2 A condensed variant of Fiat Shamir signatures

Notation. For $N \in \mathbb{N}$ let $\mathbb{Z}_N$ denote the ring of integers modulo N. The numbers t and k are security parameters, typically $4 \leq t$, $k \leq 20$.

The role of the key authentication center (KAC). The KAC chooses

- random primes p and q such that $p, q \geq 2^{256}$
- a one–way hash function $h : \mathbb{Z}_N \times \mathbb{Z} \rightarrow \{0,1\}^{tk}$.
- its own private and public key.

The KAC publishes $N = p \cdot q$, h and its public key.

COMMENTS. The KAC's private key is used for signing the public keys issued by the KAC. The KAC can use any secure public key signature scheme whatsoever for generating this signature.

The user's private and public key. Each user chooses a private key $s = (s_1, \dots, s_k)$ consisting of random numbers $s_i \in [1, N]$ such that $gcd(s_i, N) = 1$ for $i = 1, \dots, k$. The corresponding public key $v = (v_1, \dots, v_k)$ consists of the integers $v_i = s_i^{-2^t}$ (mod N) for $i = 1, \dots, k$.

Registration of users. The KAC checks the identity of a user, prepares an identification string I (containing name, address etc.) and generates a signature S for the pair (I, v) consisting of I and the user's public key v.

Signature generation.
input message $m \in \mathbb{Z}$, private key $s = (s_1, \dots, s_k)$ and modulus N.

1. *Preprocessing* pick a random $r \in [1, N]$, $x := r^{2^t}$ (mod N).
2. $e = (e_{11}, \dots, e_{tk}) := h(x, m) \in \{0,1\}^{tk}$.
3. $y := r \prod_{j=1}^k s_j^{\sum_{i=1}^t e_{ij} 2^{i-1}}$ (mod N).

Output signature (e, y).

Our signature concept reduces multicomponent signatures of the Fiat Shamir scheme to single components. The efficiency of signature generation is preserved. Step 3 can be performed as follows

$$y := \prod_{e_{t,j}=1} s_j \text{ (mod N)}$$

$$y := y^2 \prod_{e_{t-i,j}=1} s_j \text{ (mod N) for } i = 1, \dots, t-1$$

$$y := y \cdot r \text{ (mod N).}$$

Step 3 requires at most $kt + t - 1$ modular multiplications; for random e only $t(k+2)/2 - 1$ modular multiplications are required on the average. Step 1 requires t squarings and can be done in a preprocessing stage that is independent of the message m.

Signature verification.
input signature (e, y), message m, $v = (v_1, \dots, v_k), I, S, N$.

1. check the signature S for (I, v).
2. $z := y^{2^t} \prod_{j=1}^k v_j^{\sum_{i=1}^t e_{ij} 2^{i-1}}$ (mod N)
3. check that $e = h(z, m)$.

Signature verification can be done using at most $kt + t$ modular multiplications. For random e only $t(k+2)/2 + 1$ modular multiplications are required on the average. Step 2 can be performed as follows:

$$z := y^2 \prod_{e_{t,j}=1} v_j \text{ (mod N)}$$

$$z := z^2 \prod_{e_{t-i,j}=1} v_j \text{ (mod N) for } i = 1, \dots, t-1.$$

Security of signatures. In order to falsify a signature for message m the cryptanalyst has to solve the equation

$$e = h\left(y^{2^t}\prod_j v_j^{\sum_{i=1}^t e_{i,j}2^{i-1}} (\text{mod N}),\ m\right) ,$$

for e and y. No efficient method is known to solve this equation.

3 Fast Signatures

The generation of signatures can be accelerated by choosing secret keys s consisting of small integers $s_1, \ldots, s_k$. The security of this variation of the scheme is based on the assumption that computing 2^t–roots modulo N is difficult. No particular algorithms are known to compute 2^t–roots modulo N given that these 2^t–roots are of order $N^{2^{-t+1}}$.

Let the private key $(s_1, \ldots, s_k)$ consist of random primes $s_1, \ldots, s_k$ in the interval $[1, 2^{64}]$. The interval $[1, 2^{64}]$ must be large enough so that it is infeasible to find the s_i by exhaustive enumeration. We must have $t \geq 4$ so that $s_j^{2^t}$ is at least of order N^2. We next explain the requirement for the numbers $s_1, \ldots, s_k$ to be prime. For if $s_i = \alpha \cdot \beta$ with $\alpha, \beta \in [1, 2^{32}]$ we can find s_i by solving

$$\beta^{-2^t} = v_i \alpha^{2^t} (\text{mod N}) \quad \text{for } \alpha, \beta \in [1, 2^{32}].$$

This can be done using about 2^{32} steps.

For the efficiency of the scheme we suppose that $\sum_j e_{i,j} \leq 8$ for $i = 1, \ldots, t$. Then we have $\prod_{e_{i,j}=1} s_j < 2^{512}$ for all i and computing this product does not require any modular reduction. Consequently step 3 of the procedure for signature generation requires only $2t - 1$ full modular multiplications; the other multiplications are with small numbers. Thus step 3 costs an equivalent of about $2.5t - 1$ full modular multiplications. Step 1 of the procedure for signature generation requires t additional modular multiplications, but these multiplications are done in preprocessing mode independent of the message that is to be signed. The total cost of about $2.5t - 1$ modular multiplications for signature generation compares favourable with the average of $(k/2 + 1)t$ modular multiplications in the original Fiat–Shamir scheme.

4 The authentication scheme and its security

Let the private and public keys s, v be as in the previous sections. In particular we can use the small integer variant for the private key s.

The authentication protocol.
(Prover A proves its identity to verifier B)

1. *Preprocessing.* A picks a random number r with $1 \leq r \leq N$ and computes $x := r^{2^t} (\text{mod N})$.

2. *Initiation.* A sends to B its identification string I, its public key v, the KAC's signature S for (I, v) and x.

3. B checks v by verifying the signature S and sends a random string $e \in \{0,1\}^{tk}$ to A.

4. A sends $y := r \prod_j s_j^{\sum_{i=1}^t e_{i,j} 2^{i-1}} (\text{mod N})$ to B

5. B checks that $x = y^{2^t} \prod_j v_j^{\sum_{i=1}^t e_{i,j} 2^{i-1}} (\text{mod N})$ and accepts A's proof of identity if this holds.

Obviously if A and B follow the protocol then B always accepts A's proof of identity. We next consider the possibility of cheating for A and B. Let $\tilde{A}$ ($\tilde{B}$, resp.) denote a fraudulent A (B, resp.). $\tilde{A}$ ($\tilde{B}$, resp.) may deviate from the protocol in computing x, y (e, resp.). $\tilde{A}$ does not know the secret s. $\tilde{B}$ spies upon A's method of authentication.

A fraudulent A can cheat by guessing the exam e and sending for an arbitrary $r \in \mathbb{Z}_N$ the crooked proof

$$x := r^{2^t} \prod_j v_j^{\sum_{i=1}^t e_{i,j} 2^{i-1}} (\text{mod N}), \quad y := r.$$

The probability of success for this attack is 2^{-tk}.

We prove in the following theorem that this success rate cannot be increased unless we can easily compute some nontrivial 2^t–th root modulo N. For this let $\tilde{A}$ be an interactive, probabilistic Turing machine that is given the fixed values k, t, N. Let RA be the internal random bit string of $\tilde{A}$. Let the success bit $S_{\tilde{A},v}(RA, e)$ be 1 if $\tilde{A}$ succeeds with v, RA, e and 0 otherwise. The success probability $S_{\tilde{A},v}$ of $\tilde{A}$ for v is the average of $S_{\tilde{A},v}(RA, e)$, where RA, e are uniformly distributed. We assume that the time $T_{\tilde{A},v}(RA, e)$ of $\tilde{A}$ for v, RA, e is independent of RA and e, i.e. $T_{\tilde{A},v}(RA, e) = T_{\tilde{A},v}$. This is no restriction since limiting the time to twice the average running time for successful pairs (RA, e) decreases the success rate $S_{\tilde{A},v}$ at most by a factor 2.

Theorem 1. *There is a probabilistic algorithm AL which on input $\tilde{A}, v$ computes a 2^t-root of $\prod_j v_j^{c_j}$ (mod N) for some $(c_1, \ldots, c_k) \neq 0$ with $|c_j| < 2^t$ for $j = 1, \ldots, k$. If $S_{\tilde{A},v} > 2^{-tk+1}$ then AL runs in expected time $O(T_{\tilde{A},v} \,/\, S_{\tilde{A},v})$.*

Proof. The argument extends Theorem 5 in Feige, Fiat, Shamir (1987). We assume that $T_{\tilde{A},v}$ also covers the time required for B.

Algorithm with input v

1. Pick RA at random. Compute $x = x(\tilde{A}, RA, v)$, i.e. compute x the same way as algorithm $\tilde{A}$ using the coin tossing sequence RA. Pick a random $e \in \{0,1\}^{tk}$. Compute $y = y(\tilde{A}, RA, v, e)$ the same way as algorithm $\tilde{A}$. If $S_{\tilde{A},v}(RA, e) = 1$ then fix RA, retain x, y, e and go to step 2. Otherwise repeat step 1 using a new independent RA.
2. Let u be the number of probes (i.e. passes of step 1) in the computation of RA, x, y, e. Probe up to $4u$ random $\bar{e} \in \{0,1\}^{tk}$ whether $S_{\tilde{A},v}(RA, \bar{e}) = 1$. If some 1 occurs with $\bar{e} \neq e$ then compute the corresponding $\bar{y} = \bar{y}(\tilde{A}, RA, \bar{e}, v)$ and *output* $c_i = \sum_{i=1}^{t}(e_{ij} - \bar{e}_{ij})2^{i-1}$ for $j = 1, \ldots, t$ and $\bar{y}/y \pmod{\mathrm{N}}$.

Time analysis. Let $S_{\tilde{A},v} > 2^{-tk+1}$. For fixed $\tilde{A}$ and v let the success bits $S_{\tilde{A},v}(RA, e)$ be arranged in a matrix with rows RA and columns e. A row RA is called *heavy* if the fraction of 1–entries is at least $S_{\tilde{A},v}/2$. At least half of the 1–entries are in heavy rows since the number of 1–entries in non–heavy rows is at most $S_{\tilde{A},v} \cdot \#\text{rows} \cdot \#\text{columns}/2$. Thus the row RA that succeeds in step 1 is heavy with probability at least 1/2. A heavy row has at least two 1–entries.

We abbreviate $\varepsilon = S_{\tilde{A},v}$. The probability that step 1 probes $i\varepsilon^{-1}$ random RA for some $i \in \mathbb{N}$ without finding an 1–entry is at most $(1-\varepsilon)^{i/\varepsilon} < 2.7^{-i}$. Thus the average number of probes for the loop of step 1 is

$$\leq \sum_{i=1}^{\infty} i\varepsilon^{-1} 2.7^{-i+1} = O(\varepsilon^{-1}).$$

We have with probability at least 1/2 that $u \geq \varepsilon^{-1}/2$. The row RA is heavy with probability at least 1/2. If these two cases happen then step 2 finds a successful $\bar{e}$ with probability $\geq 1-(1-\varepsilon/2)^{2/\varepsilon} > 1-2.7^{-1}$, and we have $e \neq \bar{e}$ with probability $\geq 1/2$. Thus AL terminates after one iteration of steps 1 and 2 with probability

$$\geq \frac{1}{4}(1-2.7^{-1})\frac{1}{2} > 0.07.$$

The probability that AL performs exactly i iterations is at most 0.93^{i-1}. Alltogether we see that the average number of probes for AL is at most

$$O\left(5\varepsilon^{-1}\sum_{i=0}^{\infty} 0.93^{i-1} t\right) = O(\varepsilon^{-1}).$$

This proves the claim. **QED**

5 A variation of the new scheme using class groups

One can obviously modify the new scheme so that the private and public key components s_i, v_i are elements of an arbitrary finite abelian group G, i.e. we can replace the group $\mathbb{Z}_N^*$ of invertible elements in $\mathbb{Z}_N$ by the group G. The efficiency of signature generation and signature verification relies on the efficiency of the multiplication in G. For the generation of the public key components $v_i = s_i^{-2^t}$ we need an efficient division algorithm in G. The security of the authentication and the signature scheme requires that computing 2^t-th roots in G is difficult.

A particular type of suitable groups are class groups C_Δ of equivalence classes of binary quadratic forms $aX^2 + bXY + CY^2 \in \mathbb{Z}[X,Y]$ with negative discriminant $\Delta = b^2 - 4ac$. The multiplication in C_Δ, which is called *composition*, is only slightly slower than modular multiplication for integers of the order of Δ. All known algorithms for computing 2^t-th roots in C_Δ require knowledge of the group order h_Δ of C_Δ which is called the *class number.*

Class groups C_Δ have the following advantage over the group $\mathbb{Z}_N^*$:

- The problem of computing class numbers h_Δ is harder than the problem of factoring integers N of the order $N \approx |\Delta|$.
- Computing the class number h_Δ is hard no matter whether Δ is prime or composite.
- No trusted authority is required for the generation of Δ, since there is no hidden secret, as is the factorization of the modulus N in the Fiat–Shamir scheme.

For the sake of completeness we give all the details for the operation in class groups.

5.1 Class groups. A polynomial $aX^2 + bXY + cY^2 \in \mathbb{Z}[X,Y]$ is called a *binary quadratic form*, and $\Delta = b^2 - 4ac$ is its *discriminant.* We denote a binary quadratic form $aX^2 + bXY + cY^2$ by (a,b,c). A form for which $a > 0$ and $\Delta < 0$ is called *positive*, and a form is *primitive* if $gcd(a,b,c) = 1$. Two forms (a,b,c) and (a',b',c') are *equivalent* if there exist $\alpha, \beta, \gamma, \delta \in \mathbb{Z}$ with $\alpha\delta - \beta\gamma = 1$ such that $a'U^2 + b'UV + c'V^2 = aX^2 + bXY + cY^2$, where $U = aX + \gamma Y$, and $V = \beta X + \gamma Y$. Two equivalent forms have the same discriminant.
Now fix some negative integer Δ with $\Delta \equiv 0$ or $1 \bmod 4$. We will often denote a form (a,b,c) of discriminant Δ by (a,b), since c is determined by $\Delta = b^2 - 4ac$. The set of equivalence classes of positive, primitive, binary quadratic forms of discriminant Δ is denoted by C_Δ. The existence of the form $(1, \Delta)$ shows that C_Δ is non–empty.

5.2 Reduction algorithm. Each equivalence class in C_Δ contains precisely one *reduced* form, where a form (a,b,c) is reduced if $|b| \le a \le c$ and $b \ge 0$ if $|b| = a$ or if $a = c$.

5.3 Composition algorithm. The set C_Δ is a finite abelian group, the *class group.* The group law, which we will write multiplicatively, is defined as follows. The inverse of (a,b) follows from an application of the reduction algorithm to $(a,-b)$, and the unit element 1_Δ is $(1,1)$ if Δ is odd, and $(1,0)$ if Δ is even. To compute $(a_1,b_1)\cdot(a_2,b_2)$, we use the Euclidean algorithm to determine $d = gcd(a_1,a_2,(b_1+b_2)/2)$, and $r,s,t \in \mathbb{Z}$ such that $d = ra_1 + sa_2 + t(b_1+b_2)/2$. The product then follows from an application of the reduction algorithm to $(a_1a_2/d^2,\ b_2 + 2a_2(s(b_1-b_2)/2 - tc_2)/d$, where $c_2 = (b_2^2 - \Delta)/(4a_2)$.

5.4 Prime forms. For a prime number p we define the Kronecker symbol $\left(\frac{\Delta}{p}\right)$ by

$$\left(\frac{\Delta}{p}\right) = \begin{cases} 1 & \text{if } \Delta \text{ is a quadratic residue modulo } 4p \text{ and } gcd(\Delta,p) = 1 \\ 0 & \text{if } gcd(\Delta,p) \ne 1 \\ -1 & \text{otherwise.} \end{cases}$$

For a prime p for which $\left(\frac{\Delta}{p}\right) = 1$, we define the *prime form* I_p as the reduced form equivalent to (p,b_p), where $b_p = \min\{b \in \mathbb{N}_{>0} : b^2 \equiv \Delta \bmod 4p\}$.

5.5 Factorization of forms. A form (a,b,c) of discriminant Δ, with $gcd(a,\Delta) = 1$, for which the prime factorization of a is known, can be factored into prime forms in the following way. If $a = \prod_{p\,prime} p^{e_p}$ is the prime factorization of a, then $(a,b) = \prod_{p\,prime} I_p^{s_p e_p}$, where $s_p \in \{-1,+1\}$ satisfies $b \equiv s_p b_p \bmod 2p$, with $I_p = (p,b_p)$ as in 3.4. Notice that the prime form I_p is well-defined because the prime p divides a, $gcd(a,\Delta) = 1$, and $b^2 \equiv \Delta \bmod 4a$.

5.6 Choice of the discriminant and the private and public keys. We can choose $\Delta = -q$ to be the negative of any prime with $q = 3 \bmod 4$ so that q is at least 512 bits long. This particular choice of Δ implies that h_Δ is odd, and thus every class (a,b) in C_Δ has a unique square root.

We can choose the components s_i of the private key $s = (s_1,\ldots,s_k)$ to be prime forms $s_i = I_{p_i}$ with random primes p_i, $2^{63} < p_i < 2^{64}$. We must have $t \ge 3$ so that $p_i^{2^t}$ is much larger than $\sqrt{|\Delta|}$. Given s_i one can easily compute the corresponding public key component $v_i = s_i^{-2^t}$.

Acknowledgement The second author wishes to thank the Department of Computer Science of the University of Chicago for its support during this research. He also wishes to thank A. Shamir for inspiring discussions on this subject.

References

FEIGE, U., FIAT, A. and SHAMIR, A.: *Zero Knowledge Proofs of Identity.* Proceedings of STOC 1987, pp. 210 - 217, and J. Cryptology 1 (1988), pp. 469 - 472.

FIAT, A. and SHAMIR, A.: *How to Prove Yourself: Practical Solutions of Identification and Signature Problems.* Proceedings of Crypto 1986, in Lecture Notes in Computer Science (Ed. A. Odlyzko), Springer Verlag, 263, (1987) pp. 186 – 194.

GOLDWASSER, S., MICALI, S. and RACKOFF, C.: *Knowledge Complexity of Interactive Proof Systems.* Proceedings of STOC 1985, pp. 291 - 304.

GUILLOU, L.C. and QUISQUATER, J.J.: *A Practical Zero–Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory.* Proceedings of Eurocrypt'88. Lecture Notes in Computer Science, Springer–Verlag (Ed. C. G. Günther) 330 (1988), pp. 123 - 128.

MICALI, S. and SHAMIR, A.: *An Improvement of the Fiat–Shamir Identification and Signature Scheme.* Crypto 1988.

SCHNORR, C.P.: *Efficient Identification and Signatures for Smart Cards.* Proceedings of Crypto'89 (Ed. G. Brassard) Lecture Notes in Computer Science, Springer–Verlag 435, (1990) pp. 239 - 252.