The Cramer-Shoup Strong-RSA Signature Scheme Revisited

Marc Fischlin

Johann Wolfgang Goethe-University
Frankfurt am Main, Germany

marc @ mi.informatik.uni-frankfurt.de
http://www.mi.informatik.uni-frankfurt.de/

Abstract. We discuss a modification of the Cramer-Shoup strong-RSA signature scheme. Our proposal also presumes the strong RSA assumption (and a collision-intractable hash function for long messages), but —without loss in performance— the size of a signature is almost halved compared to the original scheme. We also show how to turn the signature scheme into a "lightweight" anonymous (but linkable) group identification protocol without random oracles.

1 Introduction

Cramer and Shoup [CS00] have presented a signature scheme which is secure against adaptive chosen-message attacks under the strong RSA (aka. flexible RSA) assumption, and which does not rely on the random oracle model. For a 1024-bit RSA modulus and a 160-bit (hash value of a) message a signature has about 2200 bits. Cramer and Shoup also discuss a variation of their scheme which, in addition to the strong RSA assumption, requires the discrete-log assumption and which produces signatures of roughly half the length (about 1350 bits). Here, we show that we can achieve the same signature size under the strong RSA assumption only, even with a slightly improved performance than in the original strong-RSA-only case or the discrete-log & strong-RSA case.

Our signature scheme also has the feature that for short messages, e.g., of 120 bits, a collision-intractable (or universal one-way) hash function becomes obsolete. Moreover, the signing process itself becomes slightly faster. This may be interesting for identification protocols, where users identify by signing short random messages.

At the end of this paper, we touch anonymous group identification protocols in which users can prove membership in a group without disclosing their identity. We discuss how to construct a "lightweight" anonymous (yet linkable) group identification scheme from our signature scheme. Our solution does not need random oracles, and the group's common public key as well as the performance of a single identification is independent of the number of users.

Recently, Damgård and Koprowski [DK02] have generalized the Cramer-Shoup signature scheme to generic groups. To best of our knowledge, our improvements here also apply to the model of Damgård and Koprowski.

2 A Modification of the Cramer-Shoup Protocol

In this section we recall the original Cramer-Shoup scheme, introduce our modification and prove it to be secure, and compare our proposal to the original protocol.

We adhere to the notation in [CS00]; still, the protocol description should be intelligible without [CS00]. We remark that the strong RSA assumption says that for a random RSA modulus n and a random element $z \in \mathbb{Z}_n^*$ it is infeasible to find an integer $e \geq 2$ and the e-th root of z in $\mathbb{Z}_n^*$. Hence, compared to the ordinary RSA assumption where the exponent is given, a solution for the strong RSA problem allows to come up with a self-determined exponent.

2.1 Original Cramer-Shoup Signature Scheme

The original Cramer-Shoup scheme works as follows:

Key Generation: Generate $n = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ for primes p, q, p', q'. Also pick two quadratic residues $h, x \in \mathrm{QR}_n$ and a random $(l+1)$-bit prime e'. The public verfication key is (n, h, x, e') and the private key is (p, q).

Signing: To sign a message m compute the l-bit hash value $H(m)$ with a collision-intractable hash function $H(\cdot)$. Pick a random $(l+1)$-bit prime $e \neq e'$ and a random $y' \in \mathrm{QR}_n$, compute x' where

$$(y')^{e'} = x' h^{H(m)} \bmod n$$

as well as y with

$$y^e = x h^{H(x')} \bmod n.$$

Computing this e-th root is easy given the factorization of n. The signature equals (e, y, y').

Verification: First check that e is an odd $(l+1)$-bit integer different from e', then compute $x' = (y')^{e'} h^{-H(m)}$ and verify that $x = y^e h^{-H(m)}$.

2.2 Modified Cramer-Shoup Signature Scheme

One can view the value $H(x')$ as a trapdoor commitment of the message m, using the RSA trapdoor commitment scheme. Therefore, as pointed out in [CS00], one may replace this part with any other appropriate trapdoor commitment. Indeed, [CS00, Sec. 5] suggest as an example a trapdoor commitment based on the discrete-log assumption. By this, the signature length shrinks to almost half of the original size. Unfortunately, this advantage disappears again if one switches to other trapdoor commitments based on the RSA or factoring assumption, or even general one-way functions.

The second part of the signature generation can be thought of as a representation problem. That is, a representation of x with respect to h, e, n is a pair (α, y) such that $h^\alpha y^e = x \bmod n$. In this sense, a signature in the original protocol requires that one finds a representation of x involving the hash value $-H(x')$ and a self-determined exponent e. In the modified signature scheme here, we assimilate the trapdoor commitment to the representation problem:

Key Generation: Generate $n = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ for primes p, q, p', q'. Also pick three quadratic residues $h_1, h_2, x \in \mathrm{QR}_n$. The public verfication key is (n, h_1, h_2, x) and the private key is (p, q).

Signing: To sign a message m calculate the l-bit hash value $H(m)$ with a collision-intractable hash function $H(\cdot)$. Pick a random $(l+1)$-bit prime e, a random l-bit string α and compute a representation $(-\alpha, -(\alpha \oplus H(m)), y)$ of x with respect to h_1, h_2, e, n, i.e.,

$$y^e = x h_1^\alpha h_2^{\alpha \oplus H(m)} \bmod n.$$

Computing this e-th root y from $x h_1^\alpha h_2^{\alpha \oplus H(m)}$ is easy given the factorization of n. The signature is given by (e, α, y).

Verification: Check that e is an odd $(l+1)$-bit integer, that α is l bits long, and that $y^e = x h_1^\alpha h_2^{\alpha \oplus H(m)} \bmod n$.

The idea of splitting $H(m)$ into random (but dependent) parts α and $\alpha \oplus H(m)$ is not new. It has already been applied for the well-known protocol for proving knowledge of one out of two discrete logarithms [CDS94] and for security amplification lifting random-message attacks to chosen-message attacks [CDP95]. As we will discuss below, it also gives the desired trapdoor information for proving security here.

We remark that we may instead select α at random in $\mathbb{Z}_e$ and split the message into α and $\alpha + H(m) \bmod e$. Moreover, we may alternatively define y in the signature generation as the unique value such that $x = h_1^\alpha h_2^{\alpha \oplus H(m)} y^e \bmod n$, i.e., rearrange the equation to derive a "well-formed" representation problem. Our security proof also works for these variations, even when combined.

2.3 Performance Comparison

Compared to the original scheme with signature size $2|n| + l + 1$, both the modification here as well as the one using the discrete-log trapdoor commitment produce signatures of size $|n| + 2l + 1$. Disadvantegeously, both modifications slightly increase the size of the public key, e.g., adding $|n| - l$ bits in our case.

The same speedup techniques as in [CS00, Sec. 3, 6 and 7] apply here (e.g., faster prime number generation, taking e-th roots efficiently, precomputation techniques, etc.). In particular, selecting $x = h_1^a$ and $h_2 = h_1^{a'}$ for appropriate

a, a' and storing a, a' in the secret key, the effort to compute the e-th root of $xh_1^{\alpha}h_2^{\alpha\oplus H(m)} = h_1^{\alpha+a+a'(\alpha\oplus H(m))}$ is (almost) the same as in the original scheme for $xh^{H(x')} = h^{a+H(x')}$ —and our proposal does not require the computation of the separate trapdoor commitment.

For signature verification, taking into account possible precomputations and that the discrete-log trapdoor commitment can be carried out with more efficient multiplications than in $\mathbb{Z}_n^*$ (e.g., if one works over elliptic curves or in $\mathbb{Z}_p^*$ for $|p| = 768$), the cost for the verifier for checking the single equation in our scheme seems to be comparable to the cost of checking the equation $y^e = xh^{H(x')}$ *and* the discrete-log trapdoor commitment.

Unfortunately, all solutions share the expensive prime generation of e. However, in our case, we can decrease the length of e at the cost of a larger public key. Namely, if we put, say, three values h_1, h_2, h_3 into the public key, then we can divide the hash value $H(m)$ into halves $H_1(m), H_2(m)$ of 80 bits each, and choose α and e to be 80 and 81 bits, respectively. A signature is then described by the equation $xh_1^{\alpha}h_2^{\alpha\oplus H_1(m)}h_3^{\alpha\oplus H_2(m)} = y^e$, and the signature length is about 80 bits shorter. The security proof in the next section straightforwardly extends to this case.

If we choose three generators h_1, h_2, h_3, then the effort for the signer to compute the e-th root y given stored values a, a', a'' does not change significantly in comparison to the case of two generators. But an 81-bit prime e is much easier to find than a 161-bit one. The verifier now has to perform a faster to compute "quadruple" exponentiation $h_1^{\alpha}h_2^{\alpha\oplus H_1(m)}h_3^{\alpha\oplus H_2(m)}y^e$ with 81-bit exponents instead of a "triple" exponentiation $h_1^{\alpha}h_2^{\alpha\oplus H(m)}y^e$ with 161-bit exponents.

Also note that if one wants to sign short messages in our protocol, say of 120 bits, then one can forgo the hash function H and choose e also as a shorter prime, e.g., 121 bits or even 61 bits with the trick above; it suffices that such random primes collide with negligible pobability only. For short primes this can be accomplished by using some state information like a counter.

2.4 Security Proof

We discuss that the modified signature scheme is secure against adaptive chosen-message attacks. Basically, the proof follows the one in [CS00] closely.

Note that in an adaptive chosen-message attack the adversary is given the public key of the signer and can ask the signer to sign arbitrary messages. The choice of the next message submitted to this signature oracle is adaptively determined by the data gathered before. Finally, the adversary outputs a message that has not been signed by the oracle, together with a putative signature for this message.

Let m_i be the i-th query to the signer and (e_i, α_i, y_i) denote the answer. Let m and (e, α, y) be the putative forgery of the adversary. We assume that all e_i chosen by the signer during an attack are distinct (yet, the adversary's choice e may equal some e_j), and that $H(m) \neq H(m_i)$ for all m_i (otherwise we have found a collision $m \neq m_i$).

There are two types of forgers (dubbed according to [CS00]):

Type II: The adversary outputs $e = e_j$ for some j.
Type III: The adversary outputs a new e, different from all e_i.

Type I forgers as in [CS00] disappear due to our modification. We show that type II forgers contradict the (ordinary) RSA assumption, whereas type III forgers refute the strong RSA assumption.

Type II Forger

We assume that we know j, otherwise we can guess it. Since $H(m_j) \neq H(m)$ we have $\alpha_j \neq \alpha$ or $\alpha_j \oplus H(m_j) \neq \alpha \oplus H(m)$. With probability $1/2$ we can guess in advance which case will happen, and we assume for simplicity that $\alpha_j \neq \alpha$ here. The other case is treated analogously.

We are given n, $z \in \mathbb{Z}_n^*$ and an odd prime r and are supposed to output $z^{1/r}$. To do so, we invoke the type II forger on the following public key and signature oracle: Set $e_j = r$ and for all $i \neq j$ choose a random $(l+1)$-bit prime e_i (where i is bounded by the number of queries to the signature oracle in the attack). Let

$$h_1 = z^{2\cdot\prod_{i\neq j} e_i}, \quad h_2 = v^{2\cdot\prod_i e_i}, \quad x = h_1^{-\beta} \cdot w^{2\cdot\prod_i e_i}$$

for random $v, w \in \mathbb{Z}_n^*$ and a random l-bit string β. The "prepared" public key is (n, h_1, h_2, x).

To sign the i-the message on behalf of the signer, $i \neq j$, select an l-bit string α_i and compute

$$\begin{aligned} y_i &= w^{2\cdot\prod_{k\neq i} e_k} \cdot \left(z^{2\cdot\prod_{k\neq j, k\neq i} e_k}\right)^{\alpha_i - \beta} \cdot \left(v^{2\cdot\prod_{k\neq i} e_k}\right)^{\alpha_i \oplus H(m_i)} \\ &= \left(x h_1^{\alpha_i} h_2^{\alpha_i \oplus H(m_i)}\right)^{1/e_i} \end{aligned}$$

For the j-th signature query set $\alpha_j = \beta$ and compute y_j as[1]

$$y_j = w^{2\cdot\prod_{k\neq j} e_k} \cdot \left(v^{2\cdot\prod_{k\neq j} e_k}\right)^{\alpha_j \oplus H(m_j)} = \left(x h_1^{\alpha_j} h_2^{\alpha_j \oplus H(m_j)}\right)^{1/e_j}$$

It is not hard to see that the data in this simulation is identically distributed to the one in a real attack. In particular, x and the signatures for $i \neq j$ are distributed independently of β, and therefore α_j in this simulation has the same distribution as in an actual attack.

The adversary's output yields another representation of x with respect to n, h_1, h_2 and $e_j = r$. More precisely,

$$h_1^{-\alpha_j} h_2^{-(\alpha_j \oplus H(m_j))} y_j^r = x = h_1^{-\alpha} h_2^{-(\alpha \oplus H(m))} y^r \bmod n.$$

[1] If we had bet on $\alpha_j \oplus H(m_j) \neq \alpha \oplus H(m)$ then we would have basically swapped the roles of h_1 and h_2 and would now set $\alpha_j = \beta \oplus H(m_j)$.

And, plugging in the preselected values,

$$h_1^{\alpha-\alpha_j} = h_2^{(\alpha_j \oplus H(m_j))-(\alpha\oplus H(m))} \cdot (yy_j^{-1})^r$$
$$z^{2\cdot\prod_{i\neq j} e_i\cdot(\alpha-\alpha_j)} = \left(v^{2\cdot\prod_{i\neq j} e_i\cdot((\alpha_j\oplus H(m_j))-(\alpha\oplus H(m))} \cdot yy_j^{-1}\right)^r$$

Since $|\alpha - \alpha_j| \in \mathbb{Z}_r - \{0\}$ and all e_k are relatively prime, we can compute an r-th root of z by standard procedures (see, for instance, [CS00]).

Type III Forger

This case is almost identical to the one discussed in [CS00]. Namely, given n, z preselect all e_i and set

$$h_1 = z^{2\cdot\prod_i e_i}, \quad x = h_1^a, \quad h_2 = h_1^{a'}$$

for random $a, a' \in \{1, \ldots, n^2\}$. As h_1 is a generator of QR_n with high probability and since $a, a' \bmod p'q'$ are statistically close to the uniform distribution on $\mathbb{Z}_{p'q'}$, the values x, h_2 are almost uniformly distributed quadratic residues. Also, we can sign any query m_i since we know the e_i-th roots of $xh_1^{\alpha_i}h_2^{\alpha_i\oplus H(m_i)}$ for any α_i. On the other side, the forgery yields the equation

$$y^e = xh_1^{\alpha}h_2^{\alpha\oplus H(m)} = z^m$$

where

$$m = 2 \cdot \prod_i e_i \cdot (a + \alpha + a'(\alpha \oplus H(m))).$$

The fact that $e \not| m$ with non-negligible probability and that we can compute a non-trivial $e/\gcd(e,m)$-th root of z now follows as in [CS00]. Specifically, if r is a prime dividing e, then r clearly does not divide $2 \cdot \prod_i e_i$. Write a as $a = bp'q' + c$ for $0 \leq c < p'q'$ and note that the adverary's view is essentially independent of b, even if given c. Hence, for a random choice of a the value $b \bmod r$ is almost uniform on $\mathbb{Z}_r$ and the probability that $r|(a+\alpha+a'(\alpha\oplus H(m)))$ or, equivalently, that $a + \alpha + a'(\alpha \oplus H(m)) = 0 \bmod r$ is negligibly close to $1/r$.

We conclude that with probability close to $1 - 1/r$ for the smallest prime factor r of e we have $e \not| m$. Once more, in this case it is easy to compute a non-trivial $e/\gcd(e,m)$-th root of z by standard techniques.

3 "Lightweight" Anonymous Group Identification

With an anonymous group identification scheme each user of a group is able to prove membership in the group while hiding his identity among the group members. Below, we present an anonymous group identification scheme which does not rely on random oracles, and where both the size of the group's public key as well as the computational effort for an identification are independent of the number of users in the group. Unfortunately, our protocol is linkable in the sense that a verifier is able to decide if two identifications have been carried out by the same user (although the verifier will no be able to specify the user among the group members). Also, the group manager is able to identify on behalf of any user (besides the fact that the manager can issue keys for fake users). Still, our protocol enjoys other strong security characteristics: it is for instance secure against any number of users that actively coorperate to intrude as another honest user; details follow.

Several anonymous group identification schemes (which can be derived for example from group signature schemes) have been constructed in the past, e.g., [DDP98,BF99,ACJT00,LDZ02], each with different security and performance features. Our solution seems to excell all these protocols in performance, but at the cost of unlinkability.

The group manager in our anonymous identification scheme picks an RSA modulus $n = pq$ of strong primes $p = 2p' + 1, q = 2q' + 1$, and a random element $x \in \mathrm{QR}_n$ together with a generator h_1 of QR_n. The values (n, x, h_1) make up the group's public key. If a user u wants to join, then the manager picks a random $(l + 1)$-bit prime e_u and a random l-bit value α_u, and computes y_u such that $h_1^{\alpha_u} y_u^{e_u} = x \bmod n$. The manager hands the pair (α_u, y_u) and e_u to the user.[2]

If a user u wants to identify as a group member to some verifier, both parties run Okamoto's RSA identification protocol [O92] on the user's key and the group's public key. That is, the user picks $a \in \mathbb{Z}_{e_u}$, $z \in \mathbb{Z}_n^*$ in order to calculate $A = h_1^a z^{e_u} \bmod n$ and sends this value A with e_u to the verifier.[3] The verifier answers with a random challenge $c \in \mathbb{Z}_{e_u}$ and the user conclusively transmits b, B where $b = a + c\alpha_u \bmod e_u$ and $B = zx^c h_1^{\lfloor (a+c\alpha_u)/e_u \rfloor} \bmod n$. The verifier checks that e_u is an odd $l + 1$-bit number and the correctness condition $Ax^c = h_1^b B^{e_u} \bmod n$ of the identification protocol.

Basically, our identification protocol inherits security from our signature scheme. Think of the group manager giving each new user u a signature for random message α_u. Note that this message α_u is chosen by the group manager, i.e., this setting corresponds to a random-message attack. Therefore, we do not need a trapdoor commitment nor a random splitting.

If some malicious user u^*, either a member or not, successfully identifies as another member using an exponent e_u of an honest user u, then, by the proof-of-knowledge property of Okamoto's scheme, we can extract a representation (α^*, y^*) of x with respect to e_u from this identification attempt. As Okamoto's identification is witness-independent, we have $\alpha_u \neq \alpha^*$ with probability $1 - 2^{-l}$ for the user's secret key (α_u, y_u). In this case, party u^* thus forges a signature of a new message α^* which is infeasible under the RSA assumption. Similarly, if u^* chooses a new e_{u^*} and successfully proves membership, we obtain a successful

[2] For ease of notation we switch to a "well-formed" representation problem as explained at the end of Section 2.2.

[3] Okamoto's protocol does not require to send the exponent e_u as the exponent is already part of the public key. Here, the group's public key does not contain the users' exponents, so we let the user append it to the protocol data. Indeed, this is what makes our protocol linkable.

signature forgery for message α^* for this e_{u^*}, contradicting the strong RSA assumption.

We remark that security even holds with respect to malicious users u^* who may adaptively decide to join controlled users, to corrupt existing parties, and run protocols with the honest users before trying to intrude. Using techniques developed in [BFGM01], one can even extend it to the case that u^* tries to intrude in the name of a user u while executing the identification protocol with that user u (in the presence of so-called session IDs).

Also note that we can add "threshold admittance levels" to our identification protocol almost for free. That is, each user u is assigned a privilege number ℓ_u and this user is only allowed to enter (by means of identification) level ℓ areas for $\ell_u \geq \ell$. This feature is easy to accomplish in our scheme by demanding that, in order to enter level ℓ, the user u must identify with respect to an $(l+1+\ell)$-bit (or larger) number e_u, and by letting the group manager distribute corresponding exponents to the users when joining.

Acknowledgment

We thank Ronald Cramer and Victor Shoup for comments.

References

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, G. Tsudik: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme, *Advances in Cryptology—Crypto 2000, Lecture Notes in Computer Science, Vol. 1880, pp. 255–270, Springer-Verlag*, 2000.

[BFGM01] M. Bellare, M. Fischlin, S. Goldwasser, S. Micali: Identification Protocols Secure Against Reset Attacks, *Advances in Cryptology—Eurocrypt 2001, Lecture Notes in Computer Science, Vol. 2045, pp. 495–511, Springer-Verlag*, 2001.

[BF99] D. Boneh, M. Franklin: Anonymous Authentication with Subset Queries *Proceedings of the 6th ACM Conference on Computer and Communication Security, pp. 113-119*, 1999.

[CDP95] R. Cramer, I. Damgård, T. Pedersen: Efficient and Provable Security Amplification, *CWI Reports, Computer Science, CS-R9529*, 1995.

[CDS94] R. Cramer, I. Damgård, B. Schoenmakers: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols, *Advances in Cryptology—Crypto'94, Lecture Notes in Computer Science, Vol. 839, pp. 174–187, Springer-Verlag*, 1994.

[CS00] R. Cramer, V. Shoup: Signature Schemes Based on the Strong RSA Assumption, *ACM Transactions on Information and System Security (ACM TISSEC), 3(3), pp. 161–185*, 2000.

[DK02] I. Damgr̊d, M. Koprowski: Generic Lower Bounds for Root Extraction and Signature Schemes in General Groups, *Advances in Cryptology—Eurocrypt 2002, Lecture Notes in Computer Science, Springer-Verlag*, 2002.

[DDP98] A. De Santis, G. Di Crescenzo, G. Persiano Communication-Efficient Anonymous Group Identification *Proceedings of the 5th ACM Conference on Computer and Communication Security, pp. 73-82*, 1998.

[LDZ02] C. Lee, X. Deng, H. Zhu: Desing and Security Analysis of Anonymous Group Identifcation Protocols, *Public Key Cryptography (PKC) 2002, Lecture Notes in Computer Science, Springer-Verlag*, 2002.

[O92] T. Okamoto: Provable Secure and Practical Identification Schemes and Corresponding Signature Schemes, *Advances in Cryptology—Crypto '92, Lecture Notes in Computer Science, vol. 740, pp. 31–53, Springer Verlag*, 1993.