Formal Security Proofs for a Signature Scheme with Partial Message Recovery

Daniel R. L. Brown and Don B. Johnson

February 24, 2000

Abstract

The Pintsov-Vanstone signature scheme with partial message recovery (PVSSR) is a variant of the Schnorr and Nyberg-Rueppel signature schemes. It produces very short signatures on messages with inherent redundancy. Overhead ranges from 20 to 25 bytes. This article gives a formal proof of the security of PVSSR, which reduces the difficulty of existential forgery to the difficulty of the discrete logarithm problem. The proof works in the random oracle model (which assumes an ideal hash function) combined with an ideal cipher model. Suggested instantiations for the ciphers in cryptographic applications are symmetric encryption primitives, such as DEA. A second proof is given, in which the random oracle model is replaced by the generic group model. A third proof permits the cipher to be XOR, by working in both the random oracle model and the generic group model.

1 Introduction

Signature schemes with appendix are designed to be resistant against existential forgery. In some situations, bandwidth is at a premium, and a short length of combined message and signature is needed. In some situations, the messages being signed are always of a certain form, and carry a certain amount of inherent redundancy. Signature schemes with appendix do not take advantage of such redundancy to shorten the combined length. In these situations, signature schemes with message recovery, or partial message recovery can be superior, because they can exploit inherent redundancy to shorten length.

Elliptic curve group based public key cryptographic schemes are optimal in regard to efficiency in length, and Pintsov and Vanstone [PV99] have designed a signature scheme with partial message recovery PVSSR, to be implemented with elliptic curves, to produce very short signatures. These signatures are shorter than ECDSA, DSA, and RSA based signature schemes with appendix and message recovery, with the same level of security with respect to the best known attacks.

This article proves that in certain models PVSSR is as secure as the elliptic curve discrete log problem (ECDLP) against all potential attacks, not just the known ones. The article also proves that PVSSR is secure in different models, as a form of one-wayness of hash functions. We now define PVSSR.

1.1 Signature Generation

Let $W = sG$, be the public key of a signer, where s is the private key of the signer, and G is the generator of a subgroup of an elliptic curve group of order r. Let $S_{\cdot}(.)$ be a cipher, a keyed (parameterized) family of one-to-one transformations. Let $H(.)$ be a hash function. To sign a message m, the signer breaks the message into two parts, l and n. Assume that the part n belongs to subset N, of size 2^a, of the set of binary strings of length b. Therefore the part n has effectively redundancy of b-a bits. (For example, such N could be all n that are an address, or n that are an English phrase, or n that are an executable fragment of some computer language.) The signer computes a signature (c,d) by the following sequence of steps:

1. If n is not a member of N, stop and return "invalid".
2. Randomly choose an integer u from the range $[1,r - 1]$
3. Compute $V = uG$
4. Derive a symmetric key V' from the point V using a key derivation function.
5. Compute $c = S_{V'}(n)$
6. Compute $h = H(c||l)$
7. Compute $d = sh + u \bmod r$

The security of the scheme depends on 2^{a-b} being a negligibly small probability.

1.2 Signature Verification and Message Recovery

Let W be the public of the signer, authentically obtained by the verifier. Let G,r and the elliptic curve group associated with the signer, also be authentically obtained by the verifier. Let (c,d) be a purported signature, and l a purported "verification" portion of the message m, with the given signature. The verifier verifies the signature and recovers m by the following sequence of steps:

1. Compute $h = H(c||l)$
2. Compute $V = dG - hW$
3. Derive a symmetric key V' from the point V using a key derivation function.
4. Compute $n = S_{V'}^{-1}(c)$
5. If n is not a member of N, then stop and reject the signature
6. If n is a member of N, then accept the signature
7. Recover the message as $m = l||n$

1.3 The importance of the redundancy variable, $b - a$

The number $b - a$ is the number of bits of redundancy in the message n. This is a *scalable* parameter of PVSSR, and is independent of the key length of $\log_2(r)$. For example, with key length of 160 bits, which is recommended to prevent the private key from being found by solving discrete logs, and redundancy parameter $b - a = 10$, then existential forgery is possible with probability of about 2^{-10}, or about 1 in 1000. Users of PVSSR should determine the level of resistance desired against existential forgery, based on the importance of messages being signed. Of course, more redundancy requires larger values of b, and thus longer signatures, so there is a trade-off to be decided.

In PVSSR, the choice of N is intentionally left open. For a high-speed application, the test for n Î N should be automated. If the messages being sent are such that the part n initially does not have the desired level of redundancy, it is possible to expand n, by padding, or adding redundancy by some other means. For example, it may be that there are 40 bits of natural redundancy and 40 bits of inserted redundancy, for a total of $b - a = 80$, which makes forgery roughly as difficult as extracting the private key. The source of the redundancy is not important, provided that the signer and verifier use the same N. The three proofs below are applicable for any fixed choice of N, although the third has the very plausible requirement that the cipher S be independent form N, in a sense defined in section 6.1.

1.4 Flexible Recovered Message Length

The length of the recovered part n of the message is not tied to any other parameters of the scheme. For example, the recovered portion could be 10, 20, or 100 bytes, with a 20-byte group, 20-byte hash, 8-byte block cipher (DES). There are only two requirements that affect the length of n: (1) it contains sufficient redundancy to prohibit existential forgery, and (2) if a weak one-time padding cipher such as XOR is used, then the length of n should not exceed the length of the ephemeral public key V (eg. 20 bytes). The latter requirement does not force n to be at least 20 bytes or to be padded to 20 bytes. For example, with the XOR cipher the bit string V can be truncated or folded onto itself down to a key bit string V' of the length of n. If an 8-byte block cipher is used and n slightly exceeds a block length, say with length 17 bytes, there are tricks that allow n to be encrypted and decrypted without substantial increase in length.

1.5 Elliptic curves and certificates

We recommend using elliptic curves because they offer smaller key sizes than equivalent-strength modular integer groups. For example, a 1024-bit modulus is needed to offer 80 bits of security. This does not necessarily affect the size of the signature *(c,d)*, but could be problematic because it may be necessary to include information about the public key in the form of a certificate. It is likely in that case that the need for a short signature implies the need for a short certificate. The overwhelming 128 bytes of an integer public key would eliminate the gain of message recovery. Therefore, elliptic curve groups are better suited for message recovery because of this potential problem. (In particular, implicit certificates are particularly good for such bandwidth-constrained environments, because they consist of a single point in the group, or just 20 bytes for an elliptic curve with point compression. This means that both the signature and certificate are able to fit in a limited space.)

2 Concrete Examples

2.1 Digital postage marks at 23 bytes of overhead

Digital postage marks need to convey postal data ranging from 20 to 50 bytes. Some parts of the postal data, including date, postage value and postal code of originating location are sent in the clear portion of the message *l*, for practical reasons. Other parts of the postal data, such as serial number of postage accounting device, message identification number, value of ascending register in the accounting unit, or e-mail address of the sender, are sent within recovery portion *n*. At minimum, this might include 13 bytes of data. The natural redundancy within these 13 bytes could be 7 bytes. To get 10 bytes of redundancy, 3 bytes of redundancy could be inserted by padding with 3 bytes. Then, *n* would have 16 bytes.

We recommend using a 20-byte elliptic curve (160 bits or 163 bits), SHA-1 as a good 20-byte hash function, and DES. Since *c* would have the same length of 16 bytes as *n* it does not introduce anything further overhead. The integer *d* would have 20 bytes. The total overhead is 20 bytes for *d* and 3 bytes of added redundancy, for a total of 23 bytes of overhead at 2^{-80} level of security.

2.2 Signing extremely short messages at 24 bytes of overhead

Consider signing a short 1-byte message, such as yes/no, buy/hold/sell, etc. To prevent replay attacks, such short messages often need to be sent together with a 3-byte sequence number. For the purposes of increasing forgery resistance 4 bytes of redundancy could be added. This results in an 8-byte message portion *n.* (Let *l* have 0 bytes.) With DES, SHA-1 and 20-byte EC, the signature *(c,d)* has 28 bytes, 24 of which are cryptographic overhead over the message and sequence number. The sequence number of each message has a required value, so it is redundant. Therefore, there are 7 bytes of redundancy in *n*, which gives 2^{-56} level of security against existential forgery. (Total break resistance is still at the 2^{-80} level.)

2.3 Signing and recovering longer messages at 20 bytes of overhead

If the message to be recovered is 20 bytes or longer, it is reasonable to expect that certain formatting requirements or meaningfulness of the message will result in at least 10 bytes of natural redundancy. This obviates the need to insert added redundancy. Therefore the only overhead is the 20 bytes of *d.*

(In the worst case, when the message to be recovered has zero redundancy, 10 bytes of redundancy could be added and 20 bytes for the integer *d*, for a total of 30 bytes. Ad hoc methods could be used to shave off a few bytes from the 30, but we do not recommend such methods, because it is questionable whether they have better or worse effect on security than simply adding less than 10 bytes of redundancy.)

3 Security Models

3.1 Overview of security features

The security of PVSSR depends on the security of four of its components:

1. The security of the elliptic curve group (in particular, the difficulty in the discrete logarithm problem)
2. The security strength of the hash function
3. The security of the cipher
4. The security of the set N (in particular, the smallness of 2^{a-b})

Furthermore, the security of PVSSR depends on the independence of these four components. For example, the hash function should not be defined in terms of the elliptic curve group, and the set N should not be contained in the set of all n such that $S_{V'}(n) = c$ for some fixed c.

For the sake of efficiency, some implementations may employ truncation or padding as key derivation, rather than a more secure hash-based key derivation function. The proofs in this article are valid for any key derivation function provided that it operates in conjunction with the cipher, in a secure manner.

The most desirable security proof of PVSSR would reduce its security to the security and independence of the individual components. We do not know of such a reduction. The reduction proofs given in this article work with certain models, where some heuristic properties of two components are assumed. The common principle in these heuristics is that a component is "maximally random", fully random up to being constrained by the definition of component's class (group, hash or cipher). Implementing such maximally random components is not practical. Nevertheless such proofs do provide some assurance of security for practical implementations, if the known attacks against the implemented component, whether it be the group, the hash or the cipher, are only as good as attacks against a maximally random object in the class. More details of each of the three models are given in the next subsections. We re-iterate that the three reductive proofs of this article, each work in a combination of two out of the three models.

3.2 The random oracle model

In the *random oracle model* or *ideal hash function assumption*, the hash functions invoked by a cryptographic scheme are replaced by a random oracle, that is, an algorithm with random output subject to the constraint of behaving like a function. That is, the random oracle's outputs are chosen randomly from its range of outputs, unless the input is a previous input, in which case the previous output to the input is given again. The random oracle model enables security proofs to be given for certain efficient cryptographic schemes. Such proofs are in the form of reductions of successful attacks to solutions of mathematical problems, which are conjectured to be intractable, such as the discrete logarithm problem.

The *random oracle paradigm* asserts that "secure hash functions", such as SHA-1, can securely replace random oracles in cryptographic schemes that are secure in the random oracle model. In any event, a successful attack on a scheme, secure in the random oracle model, must exploit the specific hash function used to instantiate (replace) the random oracle. No such attacks are known when using existing cryptographic hash functions, such as SHA-1.

3.3 The generic group model

In the *generic group model*, a (prime order) group used in a cryptographic scheme, is assumed to be generic, that is, representations of the elements of the group are selected randomly by some algorithm, subject only to meeting the conditions of being isomorphic to the prime-order group in question. The principle is exactly the same as the random oracle model, where the randomness is subject only to being consistent as a function. In the generic group, randomness is subject only to being consistent as a specific (prime order) group. To better model groups that can and are implemented, assume that each element of the group has a unique representation, and also that a new and random representation can be submitted to the generic group algorithm (which may subtract only, or else, may add or negate as directed).

A successful attack against a scheme, secure in the generic group model, must exploit the specific choice of group. Thus groups where some problem can be solved faster than in a generic group are not suitable for such schemes. In a generic group, the discrete logarithm problem is known to be exponentially hard. Thus families of groups where the discrete logarithm problem is not exponentially hard, are not reasonable substitutes for a generic group. Prime order subgroups of general elliptic curve groups (with secure parameters) are good examples of groups for which all known attacks against the discrete log problem are not much better than attacks in the generic group. (And the improvement, a speedup factor of the square root of 2, arises only because negatives of points are more easily computed in a elliptic curve group than sums, which distinguishes them from generic groups.)

3.4 The ideal cipher model

A new model, the *ideal cipher model* is proposed in this section. A cipher is a parameterized (keyed) family of bijective transformations. A cipher is an *ideal cipher* if it is maximally random in the sense that its values are produced by an algorithm in a way as random as possible, subject to being a consistent cipher. The algorithm may be asked to evaluate the cipher in either direction, forward or backward, that is, inverses may be computed, provided consistency and randomness are maintained.

Proposed substitutes for ideal ciphers are deterministic symmetric encryption primitives, such as DEA or AES. Such primitives are keyed families of ciphers, but have different design criteria than that of being similar to ideal ciphers. (This is in contrast to secure hash functions, where the design criteria were essentially an attempt of designing a function as unpredictable as a random oracle.) Nevertheless, some effort towards unpredictability was designed into DEA and AES, and therefore we propose to offer these cryptographic primitives as reasonable replacements for ideal ciphers for schemes proved to be secure in the ideal cipher model.

When a key derivation function is considered, the combined action key derivation and ciphering should be considered ideal. That is, for each point V in the group, the particular cipher with the key with derived from V should be as random as possible.

3.5 On reductions for specific forgers

It is worth noting, in what theoretical contexts the reductions in the following proofs are applicable. For example, suppose a signature scheme is secure in the random oracle model. Let us distinguish two categories of adversaries against the signature scheme, called forgers.

The first is the random oracle forger, and produces forged signatures provided the hash function is a random oracle. Thus, the probability of success of the random oracle forger is defined over the probability space of random hash functions. This is a very large space, and the random oracle forger is successful over a significant (non-negligible) portion of this probability space. Hence, the random oracle forger is an algorithm that is successful for a very large number of hash functions, because the largeness of the space of hash functions is very much greater than the smallness of the probability of success of the forger. The set of hash functions for which the random oracle forger is successful is so large, that it must contain some hash functions, or even pairs of hash functions, with certain special properties. These properties, together the verification operation equations, enable, either a discrete log challenge to be solved, or an RSA function to be inverted [BR96].

The second is the specific hash forger. This forger is only successful against the signature scheme, for one specific hash function. The reductions in the proofs below, at least those that assume the random oracle model are not applicable to a specific hash forger. The reductions cannot use a specific hash forger as a sub-routine in an algorithm that computes discrete logs in a specific forger. This would be desirable, but is not the case.

This limitation in regards to specific hash functions, and also specific groups and specific ciphers is addressed as follows. We provide three proofs. Each proof idealizes two of the three objects, (group, hash and cipher), by modeling these two objects by a randomized algorithm. The remaining object is permitted to be specific object, with certain security properties. In each proof, a forger in the ideal model for two of the objects is used, in a reductive algorithm, to find a flaw or weakness in the specific object. The flaws are: solving the discrete log problem in the group, solving the “concatenation problem” for a hash function, and finding “non-uniformity” in the cipher.

4 Reduction of Security to the Discrete Log Problem

4.1 The forking lemma

This section briefly describes Pointcheval and Stern’s forking lemma [PS96]. Consider a signature scheme, which invokes one evaluation of a hash function in the verification operation. For the following purposes, call this hash function evaluation the *critical hash.* Let F be an adversary to the signature scheme, which is an algorithm with input consisting of only the signer’s public key that produces signatures in the random oracle model with non-negligible probability. Adversary F is also able to query an honest signer for signatures of a sequence of messages adaptively chosen by F.

The forking lemma asserts that it is possible to use the algorithm F to obtain, with non-negligible probability, two signatures s and s' related in the following manner. The arguments to the hash function involved in the verification operation for each of s and s' are identical. The outputs of the hash function on these identical inputs are unequal with non-negligible probability. The method by which F can be used to obtain such a pair of signatures is as follows. Run the algorithm F twice. In each run, F will query the random oracle for a sequence of evaluation of hash functions. Supply identical random answers to F in each run, except for one answer, the t^{th} answer, where t is chosen at random before both runs of F. Note that before the t^{th} random oracle query, the two runs of F are identical. Therefore, the inputs to the t^{th} hash evaluation are identical in both runs of F. But, on the other hand, there is a non-negligible probability that the hash evaluated in the verification operation on the output of F, that is, the critical hash, is the same as the t^{th} random oracle query, because of two reasons. First, if F had never queried the random oracle for the critical hash, then there is negligible probability that the signature will verify. Second, F can only query the random oracle a polynomial number of times, so, since t is chosen random, and one the random oracle queries of F is the critical hash, there is a non-negligible probability that it will be the t^{th} random oracle query.

The forking lemma may appear less convincing than proofs such those in [BR96], because the forking lemma seems to require two different hash functions. In implementations, signature schemes invoke one fixed hash function. However, the forking lemma requires a random oracle forger, which is successful over many different hash functions. Thus, it is rigorous to consider two hash functions (both with outputs generated at random). The reductions in [BR96] also require a random oracle. Both types of reductions are not applicable to a specific hash forger. In this respect, the reductions in [BR96] should not be regarded as more realistic. (However, the theoretical “tightness” of the reductions is a separate issue.)

4.2 Security Proof in the Combined Random Oracle and Ideal Cipher Model

Theorem 1 *In the combined random oracle and ideal cipher model,* PVSSR, *is asymptotically secure against existentially forgery (for messages where n belongs to N) from an adaptively chosen message attack, if the discrete logarithm problem is intractable and 2^{a-b} is negligible.*

Proof (Sketch). Suppose F is an adversary that achieves forgery. With non-negligible probability, we can assume that F queries both H and S. In particular, F queries for the value of $H(c||l)$ and either $S_V(n)$ or the inverse $S_V^{-1}(c)$. Based on the order of the queries, and whether the inverse was queried, there are four cases to consider.

1. $H(c||l)$ first, then $S_{V'}(n)$. Since $S_{V'}^{-1}(c) = n$, we have $S_{V'}(n) = c$. But the value of $S_{V'}(n)$ is chosen at random, so there is negligible probability that it equals the value c (which was seen in the hash query).

2. $H(c||l)$ first, then $S_{V'}^{-1}(c)$. In this case, $n= S_{V'}^{-1}(c)$ must be chosen randomly, so the probability that n belongs to N is 2^{a-b}, which is negligible.
3. $S_{V'}^{-1}(c)$ first, then $H(c||l)$. Use Pointcheval and Stern's forking lemma technique. At random, choose an index t, and run F twice, but change the t^{th} random value of H as queried by F. Since the total number of queries is polynomial, there is a non-negligible chance that the t^{th} query of H by F is the *critical query* of H by F for the value of $H(c||l)$, in both runs of F. If h and h' are the random values returned by H in the critical queries, and (c,d) and (c',d') are the resulting signatures, then $dG – hW = V = d'G – h'W$, because the value of V was produced by F in the first query. Since $sG = W$ and $(h-h')W = (d-d')G$ it follows that $s = (h – h')^{-1}(d-d')$ mod r.
4. $S_{V'}(n)$ first, then $H(c||l)$. Use the above argument again. Note that V is still fixed, because it is determined by F before the critical query $H(c||l)$.

Thus, if F succeeds non-negligibly often, one of the latter two cases must apply. The forking lemma permits a second, similar run of F, such that the value of the discrete log s of the elliptic curve point W is found.

It remains to show how to answer the queries that F makes for signatures of messages. With knowledge of s, the signature generation algorithm can be applied, but knowledge of s is what is sought. The idea is that the ability of F to forge signatures can be used to find s. Since H and S need only be random, proceed by choosing the signature (c,d) randomly, and then answer subsequent queries of F for values of H and S in a manner consistent with this random signature. Generate (c,d) as follows:

1. Choose h, randomly from the range of H.
2. Choose d, randomly from the integers in the range $[1,r-1]$
3. Compute $V = dG – hW$
4. Compute $c = S_{V'}(n)$
5. Answer the query of F for the signature of $m=l||n$ with (c,d)

In order to be consistent, if F subsequently queries for the hash $H(c||l)$, the response must be h. Since h was chosen randomly, this complies with H being a random oracle. ❑

In the above proof, the forking lemma technique hinges on H being a random oracle. The ideal cipher model, the fact that S is "maximally" random, is exploited in a less complicated fashion.

5 Reduction of Security to the Strength of the Hash Function

5.1 Strong Hash Property

Now leave the random oracle model, and consider actual specific, arbitrary, deterministic hash function. Some of these may have a security property that we define below. This is analogous to a specific group having the property the discrete log problem is hard.

Definition 1 (Strong Hash) *Let H be a hash function. If there does not exist a probabilistic polynomial time algorithm A which first finds a value h or l_0 and then finds, on random input c, some l such that $H(c || l) = h$ or $H(c||l) = H(c||l_0)$, with non-negligible probability, then H is a strong hash function.*

We call the problem of finding such l the target value problem and target collision problem. If the value c is regarded as a key for a family of hash functions $H(c,\cdot)$, then collision part of the above hash strength is called *target collision resistant (TCR)* by Bellare and Rogaway [BR97], and is equivalent to Naor and Young's universal one-way security, and has also been called 2nd-preimage-resistance and weakly collision-resistance. The other part of the above hash strength, we call *target value resistance (TVR)* to correspond to [BR] terminology TCR, but it has also been call one-way-ness, and preimage-resistance. In effect, "strong = TCR + TVR" (where the prefix is regarded as a "key" or "index"). We propose that SHA-1 is a good candidate for a strong hash function.

5.2 Observable combination argument

In a generic group of order r, where r is a prime, the *observable combination argument*, adapted from Shoup [S98], is the following. Let A be any algorithm which starts with representations of two points, G and W, and subsequently in its operation, submits queries to the generic group. Assume that A makes only a number of queries that is polynomial in log r. If V is any representation, seen either as the input or output by the group oracle, then either V is an *observable* integer combination of G and W, say $V = xG+yW$, or V is the representation of some point in the group, which could any almost any point, that is, over the random space of choices made by the generic group algorithm, $V=uG$, where u is nearly uniformly distributed from the integers in the range [1, $r-1$], and is nearly independent of s, where $W = sG$. Observable here means that V is either G or W, or was the output of the generic group algorithm in response to a query for the sum, difference or negation of observable representations of points. If A chooses a new representation (neither G,W or any past outputs) to input to the generic group algorithm, then neither it, nor the output is observable (as a combination of G and W).

Shoup [S98] demonstrated that there is no such algorithm A as above that can find s such that $W=sG$, in time less than $O(r^{1/2})$. We use this fact in the proof below.

5.3 Security Proof in the Combined Ideal Cipher and Generic Group Model

Theorem 2 *In the combined generic group and ideal cipher model,* PVSSR, *is asymptotically secure against existentially forgery (for messages where n belongs to N) from an adaptively chosen message attack, if the hash function H is strong and* 2^{a-b} *is negligible.*

Proof (Sketch). Suppose F is an adversary that produces forged signature (c,d) with corresponding verification message portion l. With non-negligible probability, we can assume that F queries both the generic group and S, because otherwise there is negligible probability that the signature will be accepted by the verification operation. In particular, the representation $V=dG - H(c||l)W$ must appear as the input or the output of a query to the generic group algorithm and either the query $S_{V'}(n)$ or query of for the inverse $S_{V'}^{-1}(c)$ must be made. The order and the nature of the queries, leads to the following cases:

1. Suppose that $S_{V'}(n)$ or $S_{V'}^{-1}(c)$ was queried before V appeared as the representation of a point in the context of the generic group algorithm. Then, there is negligible chance that V first appears as an output of the generic group, so V must be an input to the generic group algorithm. Thus V is not observable. By the observable combination argument, $V = uG$, where u is almost uniformly distributed. But F finds c,d,l such that $uG = dG - H(c||l)W$, which fixes u at a particular (albeit unknown) value $u=s+H(c||l)s$, which is not independent of s. This is a contradiction.
2. Suppose that V was queried by F to the generic group algorithm as an input, before the query $S_{V'}(n)$ or $S_{V'}^{-1}(c)$. Then again, V is not observable, so the above argument applies. If V is not observable, as a result of being the output to query with a new representation input, then again V is not observable, and the above contradiction results.
3. Suppose that V appeared as the observable output of a query by F to the generic group algorithm, prior to the F's query $S_{V'}(n)$ or $S_{V'}^{-1}(c)$. Suppose the latter query was $S_{V'}^{-1}(c)$. The response n, which is chosen randomly, has a negligible probability of falling into N, which is a contradiction.
4. Suppose that V appeared as the observable output of a query by F to the generic group algorithm, prior to the F's query $S_{V'}(n)$ or $S_{V'}^{-1}(c)$. Suppose the latter query was $S_{V'}(n)$. Since V is observable, $V = gG+hW$ for some integers g and h. Choose the response $c=S_{V'}(n)$ randomly, as required. Then F finds d,l such that $V = dG - H(c||l)W$.
   a) If $(g,-h)$ ^{1} $(d,H(c||l))$, then solve for $s=(h+H(c||l))^{-1}(d-g)$ mod r, which contradicts Shoup's result that that s cannot be found in polytime. (That is, the discrete log problem is exponentially difficult in the generic group model.)
   b) If $(g,h) = (d,H(c||l))$, and V is not an ephemeral key of a signing query, then F has first found h, and then found, for random c, an l such that $H(c||l) = h$, which contradicts the assumption that H is strong, because H is TVR-broken.
   c) If $(g,h) = (d,H(c||l))$, and V is the an ephemeral key of a signing query of message $l_0||n_0$, then F has first found l_0, and then found, for given random c, an l such that

$H(c||l) = H(c||l_0)$, which contradicts the assumption that H is strong, because H is TCR-broken.

In all the above cases, there is only a negligible chance of success for F, so no F with non-negligible chance of success exists, under the given models and assumptions.

It remains to show how the queries of F for signatures can be answered. With knowledge of s, the signature generation algorithm can be applied, but knowledge of s is what is sought. The idea is that the ability of F to forge signatures can be used to find s. Since the group and S need only be random, proceed by choosing the signature (c,d) randomly as below, and then answer subsequent queries of F for values of H and S in a manner consistent with this random signature. Generate (c,d) as follows:

1. Choose c, randomly from the range of S
2. Compute $h = H(c||l)$.
3. Choose d, randomly from the integers in the range $[1,r-1]$
4. Compute $V = dG – hW$
5. Answer the query of F for the signature of $m=l||n$, with (c,d)

In order to be consistent, if F subsequently queries for the cipher from key V, n, the response must be c, and vice versa. Since c was chosen randomly, this complies with S being an ideal cipher. ❑

6 Reduction of Security to the Cipher Strength

6.1 Uniform decipherment property

The third proof works in the combined generic group and random oracle model, in order to reduce the security of PVSSR to the strength of the cipher. Thus, rather than work in an ideal hypothetical model where the cipher S is chosen from a random space, assume that the specific implemented cipher S (together with the key derivation function) has the following very plausible property with respect to the set N of redundant message portions.

Definition 2 (Uniform Decipherment) *Let S be a cipher (including a key derivation function). Let N be a subset of size* £2^a *of binary strings of length b. Then cipher S is uniform with respect to N if for each fixed value of c, the probability over random V that* $S_V^{-1}(c)$Î N *is* $O(2^{a-b})$. *If it is infeasible to find c such that* $S_V^{-1}(c)$Î N *with chance significantly greater than* 2^{a-b}, *then S has weak uniform decipherment with respect to N.*

In other words, S is uniform with respect to N, if there does not exist a ciphertext which deciphers to plaintext in N with much higher probability than expected over the random space of keys V. If $S = DEA$ and N is ASCII encoding of English text, this type of uniformity is plausible. Indeed, for $S = XOR$, uniformity is true if $b \leq$ the key-length of V. If the key-space of S is smaller than 2^b then, for each c, the set $N_c = \{ n \mid n = S_V^{-1}(c)$ *for* V *in the key-space of* S $\}$ is such that S is not uniform with respect to N_c because the probability in Definition 2 is 1, which is not $O(2^{a-b})$.

Therefore, unlike the previous two proofs, the following security proof is applicable for $S = XOR$, provided the key-lengths are appropriate. Use of XOR enables greater time-efficiency, so the following security proof is a desirable assurance for those implementations needing the speed of XOR.

(If a cipher S is assumed to be pseudorandom, as is suggested for DEA in [BR97], then S has weak uniform decipherment for all N for which membership can be efficiently determined. Suppose otherwise: that S is pseudorandom, but S does not have weak uniform decipherment with respect to N. Then some c can be found such that S and a truly random cipher R can be distinguished as follows. Let $f=S_V$ or else f be some random permutation (generated by R), where the choice is unknown to the distinguisher. If $f^{-1}(c)$Î N, the distinguisher guesses that $f=S_V$ and otherwise guesses that f was generated by the truly random cipher. The distinguisher has a good chance of being correct, because if f was chosen randomly, then $f^{-1}(c)$Î N with probability 2^{a-b}, which is much smaller than the probability that $S_V^{-1}(c)$Î N.)

6.2 Security Proof in the Combined Random Oracle and Generic Group Model

Theorem 3 *In the combined random oracle and generic group model,* PVSSR, *is asymptotically secure against existentially forgery (for messages where n belongs to N) from an adaptively chosen message attack, if the cipher S has (weak) uniform decipherment with respect to N and 2^{a-b} is negligible.*

Proof (Sketch). Suppose F is an adversary that produces forged signature (c,d) with corresponding verification message portion l. With non-negligible probability, we can assume that F queries both the generic group and the random oracle (hash function) because otherwise there is negligible probability that the signature will be accepted by the verification operation. In particular, the representation $V=dG - H(c||l)W$ must appear as the input or the output of a query to the generic group algorithm, and the hash query for the value $H(c||l)$ must be made. The order and the nature of the queries, leads to the following cases:

1. Suppose that V was not an observable integer combination of G and W. That is, V is an input or an output to query by F to the generic group algorithm, in which of the inputs supplied by F, was distinct from previous representations of points processed by the generic group algorithm. Note that $V = uG$ for some u, and $V = dG - H(c||l)W$, according to the verification operation. This means that $u = d + H(c||l)s$, which contradicts the observable combination argument that u is almost uniformly distributed and independent of s
2. Suppose that V is an observable integer combination, where it can be observed that $V = gG-hW$, at the time V is first processed by the generic group algorithm. Suppose that (g,h) ¹ $(d,H(c||l))$. Then $s = (h-H(c||l))^{-1}(d-g) \bmod r$, which contradicts the fact that the discrete log cannot be solved in the generic group in polynomial time.
3. Suppose that V is an observable integer combination, where it can be observed that $V = gG-hW$, at the time V is first processed by the generic group algorithm. Suppose that $(g,h) = (d,H(c||l))$. Suppose that the hash query $H(c||l)$ occurs after the first observation of V. Then, there is negligible probability that $h = H(c||l)$.
4. Suppose that V is an observable integer combination, where it can be observed that $V = gG-hW$, at the time V is first processed by the generic group algorithm. Suppose that $(g,h) = (d,H(c||l))$. Suppose that the hash query $H(c||l)$ occurs before the first observation of V. Then, the representation V is chosen randomly by the generic group algorithm, but is such that $S_{V'}^{-1}(c)$Î N, with non-negligible property. Thus, F has found a value of c that demonstrates that S is not uniform with respect to N, which is a contradiction.

In all above cases, there is only a negligible chance of success for F, so no F with non-negligible chance of success exists, under the given models and assumptions.

It remains to show how the queries of F for signatures can be answered. With knowledge of s, the signature generation algorithm can be applied, but knowledge of s is what is sought. The idea is that the ability of F to forge signatures can be used to find s. Since H and S need only be random, proceed by choosing the signature (c,d) randomly, and then answer subsequent queries of F for values of H and group operations in a manner consistent with this random signature. Generate (c,d) as follows:

1. Choose h, randomly from the range of H.
2. Choose d, randomly from the integers in the range $[1,r-1]$
3. Compute $V = dG - hW$
4. Compute $c = S_{V'}(n)$
5. Answer the query of F for the signature of $m=l||n$ with (c,d)

In order to be consistent, if F subsequently queries for the hash $H(c||l)$, the response must be h. Since h was chosen randomly, this complies with H being a random oracle. ❑

7 Practical (Traditional) Security

Provable security concerns *potential* adversaries. This is a good basis on which to compare new schemes such as PVSSR. But it is also important to consider *known* adversaries, when selecting parameters for an implementation of the scheme. Let us call this *practical security.* In other words, in practical security, one selects parameters that make the best known attacks infeasible. Generally, if possible, the best known attacks should be even more infeasible than merely potential attacks. Greater significance is given to the practical security, as more time and effort is spend on its cryptanalysis, because the "best" in "best known attack" becomes stronger. Therefore, to initiate the knowledge set of actual attacks on which to base the practical security we identity three attacks.

Each of the three security assumptions is necessary for any implementation of PVSSR. If any of the cipher, the group or the hash is known to fail to meet its security assumption, then forgery of the implementation PVSSR is immediate from this security flaw. More precisely, the weak uniform decipherment, difficulty of the discrete logarithm problem and a strong hash function (especially a TCR hash function), are necessary for a secure implementation of PVSSR. Indeed, a security flaw in one of three is sufficient for an attack that is generic for the other two components, no matter how "secure" they are. Because the attacks identified above are based on the same assumptions that security proofs are based, it can be concluded that the assumptions in the security proofs cannot be weakened. Thus the provable security is qualitatively tight. The degree of quantitative tightness must await an exact security analysis. Most likely, in the proof of Theorem 1, which requires the forking lemma, there is quantitative decrease in the security, dependent on the number of queries permitted to the hash oracle. But any gap between the provable security and practical security is essentially quantitative.

From the perspective of practical security, the exponential difficulty of the ECDLP, and present-day difficult of ECDLP, certainly offsets the possible quantitative gap between the provable and practical security of PVSSR. Indeed, while no such gap may exist for IFSSR, the basic IF problem is hindered by present-day inefficiency for some applications, and sub-exponential solutions to the IF problem for future applications, that we think PVSSR is certainly warranted as a scheme that has robust at least that of IFSSR. It may be true that sub-exponential attacks against ECDLP could be discovered, but the practical security of the ECDLP is certainly catching up to the IF, as the amount of effort put into cryptanalyzing each problem is equalizing. The difficulty of ECDLP is gaining acceptance, and has withstood more time relative to its introduction into cryptography without the discovery of sub-exponential than the IF problem did. In all, EC-PVSSR is efficient, practically secure and provable secure.

8 More about Bit Lengths and Security Recommendations

8.1 Examples of bit lengths

For a concrete example of bit lengths in PVSSR, consider the following. Use a standard elliptic (with secure parameters) of 160-bits. Then, it takes roughly 2^{80} bit operations to solve DLP in this group (and thus to directly find the signer's private key). Suppose that we wish to sign a message m of length $mlen \leq 64$ bits and redundancy $rlen$ bits. Pad out m with $64 - mlen$ zero bits. Break up the message as $m=l||n$, where $l = e$, the empty string, and $n = m$. Then n has redundancy $64 - mlen + rlen$. If $mlen \leq rlen + 14$, then n will have at least 40 bits redundancy, which makes existential forgery very difficult. (For exmaple, assuming that the original message m is an English phrase encoded in ASCII having 6 bits of redundancy for every 8 bits, then $mlen \leq 56$ suffices to ensure 40 bits of redundancy in n.) Choose a cipher DEA, and KDF1 for key derivation, which uses a hash SHA1 to obtain a 56-bit string from an elliptic curve point. The signature length is the length of (c,d) which is 224 bits. In comparison, the total length of the message m and an ECDSA signature on m, with a 160-bit EC curve, would be $320 + mlen$ bits.

For a high-speed implementation with $rlen \geq 40$, and $mlen$ any non-negative integer, use $S = XOR$, and derive the key V' rapidly as follows. Let $n=m$ and $l=e$ (do not pad n). If $mlen \leq 160$, partition the bit string representation of V into segments of length $mlen$, padding the last segment with zeros. Then XOR all the segments together to form V'. If $mlen \geq 160$, then take the first $mlen$ bits of $V||V||V||V||\ldots$ to be the derived key V'. Such operations, or variations of, can be done very quickly, and the uniform decipherment property remains highly plausible, for N being human language. The proofs in the ideal cipher model (the first and second theorems) carry less weight for the security of such an implementation.

8.2 Security Recommendations

The three security proofs of this article, each work in two ideal models and one security assumption. An implementation of PVSSR should, as much as is practical, use a group, hash and cipher, for which the security assumptions hold, are believed to hold because they have tried and tested. Specifically, the discrete log problem should be hard in the group, the concatenation problem should be hard for the hash, and the uniform decipherment property should hold. These are the *primary* security requirements.

It is also recommended that if a particular group, hash or cipher is found to have any significant "security" flaws in another cryptographic context, even if unrelated to the primary security requirements for PVSSR as described above, then it should be rejected for PVSSR. The grounds for rejection are that the flaws make it an unsuitable replacement for an ideal object, because it is not sufficiently random or unpredictable. Of course, no particular object is ideal, but caution dictates that, the existence of attacks better that work much better than attacks on an ideal version of the object, means the objects should be avoided, even if the attacks are unrelated to the attacks on PVSSR. For example, groups with index calculus attacks on the discrete log problem may be rejected, because index calculus attacks are not possible in a generic group. Therefore, the proofs relying on the generic group carry less weight for implementations of PVSSR using such groups. If collisions can be found too easily in a certain hash function *H*, then another hash function should considered. This is recommended even if there is no known attack on the concatenation problem for *H*.

9 Conclusion

Three proofs were given for the security of PVSSR. Provided that PVSSR is implemented with a (elliptic curve) group, hash function, and cipher which do not fail to meet, to the best current knowledge, certain specific security requirements, the three security proofs offer good security. In particular, the three proofs eliminate the existence of attacks against PVSSR which are successful over all possible choices for two of a list of objects (group, hash and cipher), and a single specific but secure choice for remaining member of the list (group, hash or cipher). Thus, it is possible to conclude that any attack on PVSSR with a secure choice of group, hash and cipher, must depend in some specific way on at least one member of the list group, hash and cipher. Such specific attacks against PVSSR can be prevented, by replacing the offending group, hash or cipher. For example, index calculus to find the signer's private key, is an attack that depends on the specific choice of group, the multiplicative group of integers modulo a prime. But the attack does not work if the implemented group in the PVSSR is an elliptic curve group, because index calculus is not defined.

Furthermore, given the assumptions on the specific choice of cipher, group and hash in an implementation of PVSSR, the only threats not protected against by the three theorems are from adversaries that are specific to two of the three of components, the cipher, group or hash. Provided that the chosen components are designed independently of one another, it seems very implausible that such a specific adversary could exist (without being a special case of a more generic adversary). For example, it would be a remarkable coincidence, if a relation were discovered between a standard cryptographic elliptic curve group and SHA1 to permit forgery of the associated implementation of PVSSR.

References

[BR96] M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures - How to Sign with RSA and Rabin,* Eurocrypt 1996.

[BR97] M. Bellare, P. Rogaway, *Collision-Resistant Hashing: Towards Making UOWHFs Practical,* Advances in Cryptology – CRYPTO 97, Spring-Verlag, 1997.

[BDJR97] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, *A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation,* Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997.

[PS96] D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes,* Eurocrypt 1996.

[PV99] L. A. Pintsov and S. A. Vanstone, *Postal Revenue Collection in the Digital Age,* Financial Cryptography, 2000.

[S98] V. Shoup, *Lower Bounds for Discrete Logarithms and Related Problems,* 1998.