Formal Security Proofs for a Signature Scheme with Partial Message Recovery

Daniel R. L. Brown and Don B. Johnson

June 14, 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 intrinsic redundancy. At 80 bits of security, cryptographic overhead (message expansion) ranges from 20 to 30 bytes, depending on the amount of intrinsic redundancy in the message being signed. (In comparison, an ECDSA signature with the same domain parameters would have an overhead of about 40 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 3DES or AES. 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

Several signature schemes with appendix, such as DSA, ECDSA, and those based on RSA and Rabin-Williams, are considered to be both computationally efficient and heuristically or provably secure against existential forgery by adaptive chosen message adversaries. However, when bandwidth is at a premium, a potential problem with signature schemes with appendix is that the combined length of the message and signature is too long. Signature schemes with total or partial message recovery provide a solution to this problem by embedding all or part of the message within the signature. (A secondary feature of using message recovery is that all or part of the message is "hidden" because it cannot be recovered without the public key of the singer.)

Signature schemes with message recovery specify that some message representative is recovered from the signature. For verification to be complete, the message representative must have the correct redundancy. Roughly speaking, redundancy means that the message representative belongs to a particular small subset of all possible bit strings that can be recovered from candidate signature data. Signature

schemes with message recovery generally do not specify the form of the redundancy. Typically, the form of redundancy depends on the application.

One form of redundancy is padding. For example, the recovered message representative could be required to have 80 bits of a specific padding. To sign a message it is first padded with 80 bits and then the signature is generated. However, the added bandwidth of such a full padding method can negate the bandwidth advantage of using message recovery. A more appropriate choice of redundancy for message recovery is to use the intrinsic redundancy of messages that are signed.

In typical applications, messages that are signed belong to a "meaningful" subset of messages. In other words, they have intrinsic redundancy. Thus, message recovery is particularly advantageous for applications that only use messages where the intrinsic redundancy can be precisely and easily specified and verified. For example, in a digital postage mark, certain postage information must always be present and constitutes the necessary intrinsic redundancy. If the application-specific intrinsic redundancy is insufficient for the required security level, then the intrinsic redundancy may be combined with added redundancy in the form of padding.

Some signature schemes with total or partial message recovery have restrictions on the length of the message representative to be recovered. For example, in the Nyberg-Rueppel scheme, the length is fixed. This restriction has two disadvantages. First, for very short messages, the fixed length message representative contains more redundancy than necessary and thus wastes bandwidth. Second, messages that are slightly too long may not fit within the space provided for the message representative. It would be preferable to use a signature scheme with message recovery without this restriction, because the scheme would be then be usable with a wider class of applications.

The Pintsov-Vanstone Signature Scheme with Recovery (PVSSR) is a signature scheme with partial message recover without restriction on the message representative length. When used with elliptic curves, it provides very short bandwidth. For example, at a security level of 80 bits, the cryptographic overhead of an elliptic curve PVSSR signature is 160 bits plus the number of bits of padding redundancy. In comparison, an ECDSA over the same elliptic curve domain parameters would have an overhead of about 320 bits.

(Optionally, PVSSR's flexibility allows the total amount of redundancy (intrinsic plus padding) to be set quite low in order to save bandwidth. This provides for very low bandwidth as a tradeoff for security against forgery. Although this low redundancy mode compromises the resilience against forgery, it does not seem to compromise the private key of the signer. Therefore, for messages of low importance, low redundancy PVSSR signatures could be useful for their bandwidth efficiency, without compromising the private key of the signer.

This optional feature is an another advantage of PVSSR, but this article will not directly pursue any further security analysis of PVSSR used in this mode.)

The PVSSR is scheme is an adaptation of the Nyberg-Rueppel signature scheme with message recovery. Since it a relatively new scheme, thorough cryptanalysis has not yet established is security. To establish the security of PVSSR, this article provides three separate proofs of security for PVSSR. The first proves that in certain models PVSSR is as secure as the elliptic curve discrete log problem (ECDLP). The second proves in certain models that PVSSR is as secure as the degree of a special form of collision-resistance and one-way-ness of a hash function. The third proves in certain models, that PVSSR is as secure as a certain relationship between a cipher and the selected choice of redundancy.

2 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 divides the message into two parts, l and n, according to application dependent criteria. The message 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 all n that are an English phrase, or all n that are an executable fragment of some computer language. If necessary, the message part n can be created by the signer using some message data and padding it with a certain number of bits. (The verifier removes the padding bits after recovering n.) 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 resulting signature (c, d) must be conveyed to the verifier. The verifier also need l to recover n.

2.1 Signature Verification and Message Recovery

Let W be the public of the signer, authentically obtained by the verifier. (Assume that the domain parameters for W, including G, r and the elliptic curve group, are also authentically obtained by the verifier.) Assume that the verifier authentically obtains means to test for membership in the redundancy space N. 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 $m = l||n$

If n includes padding that the signer has added, then the verifier will remove the padding from n. The padding method should be unambiguous and its quantity and form authentically pre-established between the signer and verifier. The security of the scheme depends on 2^{a-b} being a negligibly small probability.

2.2 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 $n \hat{I} N$ 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 first two proofs in this paper are applicable for any fixed choice of N while the third has the

requirement that the cipher *S* be "independent" from *N*, in a sense defined in section 7.1.

2.3 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 (such as 3DES). There are two main requirements that affect the length of *n*. First, it must contain sufficient redundancy to prohibit existential forgery. Second, 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* (e.g. 20 bytes). (This is not a requirement that *n* be at least 20 bytes or 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*. However, if a mask generation function is to derive a mask from *V* of length *n*, then the length of *n* is not directly bounded by the length of *V* but rather by certain security properties of the mask generation function. If an 8-byte block cipher, such as 3DES, 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.

2.4 Elliptic curves and certificates

We recommend using elliptic curve arithmetic because of the resulting smaller public key sizes than equivalent-strength ordinary integer discrete log based modular arithmetic. For example, for the ordinary discrete log case, the public key should be 1024 bits to give about 80 bits of security. Although, the public key size does not necessarily affect the size of the signature *(c, d)*, signatures are often sent together with a certificate containing the public key of the signer. If certificates are needed, it is likely in that case that the need for a short signature implies the need for a short certificate. The more than 1024 bits of an integer public key would eliminate the bandwidth efficiency gained from message recovery. Therefore, elliptic curve groups are perhaps best suited for message recovery.

(Incidentally, as noted in [PV99], using implicit certificates instead of ordinary certificates saves even more bandwidth. In implicit certificates, a single point in the group, 163 bits defining a compressed elliptic curve point, replaces a public key and a digital signature (totaling to at least 480 bits if an ECDSA signature is used). This means that both the signature and the certificate are able to fit in a limited space.)

3 Concrete Examples

3.1 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 3DES. 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.

3.2 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 padding 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 constitute the 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.)

3.3 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.)

4 Security Models

4.1 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 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.

Ideally, a 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.

4.2 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 typically reductions from successful attacks on the scheme to solutions of difficult, mathematical problems, such as the discrete logarithm problem, which are conjectured to be intractable.

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.

4.3 The generic group model

In the *generic group model*, a (prime order) group used in a cryptographic scheme, is assumed to be a generic group. In a generic group, 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 similar to 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 actual cryptographic groups that are implemented, we assume that in a generic group each element of the group has a unique representation. Furthermore, we assume that is possible to submit new arbitrary representations as input to the generic group operations.

If a scheme is secure in the generic group model, then a successful attack against the scheme must exploit the specific choice of group (i.e. process the group by means other than by invoking its operations as an oracle). In a generic group, the discrete logarithm problem is known to be exponentially hard [S98]. 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 an elliptic curve group than sums, which distinguishes them from generic groups.)

Certain elliptic curve groups have been standardized by NIST (see [N99]. Hopefully the discrete logarithm problem for these groups will remain about as difficult as the discrete logarithm problem for generic groups of similar order.

4.4 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 oracle that evaluates the cipher may be asked to evaluate the cipher in either direction, forward or backward. In other words, inverses may be computed, provided that the consistency and randomness of the functions are maintained.

Proposed substitutes for ideal ciphers are deterministic symmetric encryption primitives, such as 3DEs or AES. Such primitives are keyed families of ciphers, but have different design criteria than that of being similar to ideal ciphers. Nevertheless, some effort towards unpredictability was designed into 3DES 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 to be ideal. In other words, for each point V in the group, the particular cipher with the key V' derived from V should be as random as possible.

5 Reduction of Security to the Discrete Log Problem

5.1 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. 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.)

5.2 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.

$H(c||l)$ 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).

1. $H(c||l)$ 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.
2. $S_{V'}^{-1}(c)$ 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 $s = (h - h')^{-1}(d-d') \bmod r$.

3. $S_{V'}(n)$ 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.

6 Reduction of Security to the Strength of the Hash Function

6.1 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. H is a strong 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.*

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,·)*, 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 called one-way-ness, and pre-image-resistance. In other words, "strong = TCR + TVR". We propose that SHA-1 is a good candidate for a strong hash function.

6.2 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*, in the following sense: *V* is either *G* or *W*, or was the output of the generic group algorithm in response to a query by *A* for group operation on two previous observable representations.

or

- *V* could be the representation of almost any point in the group. In other words, 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*.

If *A* chooses a new representation *V* (neither *G,W* or any past outputs) to input to the generic group algorithm, then neither *V*, nor the output given by the oracle is "observable".

Shoup [S98] essentially 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.

6.3 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 $(g,-h) \cdot (d,H(c||l))$, then solve for $s=(h+H(c||l))^{-1}(d-g) \bmod r$, 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. ❑

7 Reduction of Security to the Cipher Strength

7.1 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 $S_V^{-1}(c) \hat{I} N$ is $O(2^{a-b})$. If it is infeasible to find c such that $S_V^{-1}(c) \hat{I} N$ 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* = 3DES 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* provides 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 3DES 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 $f^{-1}(c) \hat{I} N$, 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 $f^{-1}(c) \hat{I} N$ with probability 2^{a-b}, which is much smaller than the probability that $S_V^{-1}(c) \hat{I} N$.)

7.2 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 $(g,h) \neq (d,H(c||l))$. 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 $S_{V'}^{-1}(c) \hat{I} N$, 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 $[1,r\text{-}1]$
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. ❑

8 Necessary Security Requirements

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. Thus the weak uniform decipherment property, difficulty of the discrete logarithm problem and a strong hash function (especially a TCR hash function) are each necessary for a secure instantiation of PVSSR. A security flaw in one of the three components is sufficient for an attack against PVSSR, even if other two components are perfectly "secure". 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 (although the heuristic models could be weakened). The three proofs of this article find necessary and sufficient conditions for the security of PVSSR, at least heuristically and qualitatively. Further work could improve on these proofs by using less heuristic models and quantitative analysis in the reductions.

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.

[N99] National Institute of Standards and Technology, www.nist.gov/encryption, 1999.

[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*, Proceedings of the Fourth Annual Conference on Financial Cryptography 2000, (to appear).

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