Analysis of the SPV Secure Routing Protocol

Anton Mityagin, Saurabh Panjwani, and Barath Raghavan
University of California, San Diego
{amityagin,panjwani,barath}@cs.ucsd.edu

ABSTRACT

We analyze a secure routing protocol, Secure Path Vector (SPV), proposed in *SIGCOMM 2004*. SPV aims to provide authenticity for route announcements in the Border Gateway Protocol (BGP) using an efficient alternative to ordinary digital signatures, called constant-time signatures. Today, SPV is widely considered the best cryptographic defense for BGP.

We find subtle flaws in the design of SPV which lead to attacks that can be mounted by 60% of Autonomous Systems in the Internet. In addition, we study several of SPV's design decisions and assumptions and highlight the requirements for security of routing protocols. In light of our analysis, we reexamine the need for constant-time signatures and find that certain standard digital signature schemes can provide the same level of efficiency for route authenticity.

1. Introduction

Today's Internet relies upon a single protocol, the Border Gateway Protocol (BGP), for wide-area route propagation. Each autonomous system (AS) summarizes its network as a set of IP prefixes and uses BGP to inform other ASes of how to route to these IPs. Numerous documented [18] and undocumented attacks against BGP highlight the need for routing security. Ideal solutions to secure BGP should protect against all known attacks, be incrementally deployable, and impose minimal computational and communication overhead for BGP-speaking routers. Of particular interest are protocols to provide *route authenticity*—a guarantee that all routes advertised by all ASes correspond to real paths through the network learned through route announcements.

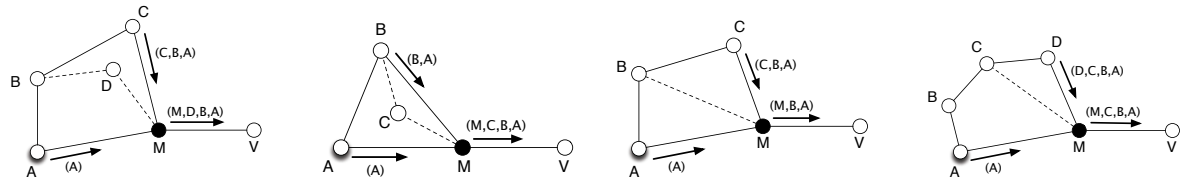
In *SIGCOMM 2004*, Hu, Perrig, and Sirbu proposed a mechanism for securing BGP with these design constraints in mind [12]. Their protocol, *Secure Path Vector (SPV)*, is unique in that it aims to simultaneously provide strong route authenticity guarantees and high performance—this is in contrast to several other proposals for securing BGP that are either computationally intensive [13] or do not make strong claims about security [22, 23, 24]. Due to these qualities, SPV has enjoyed substantial interest both from the research community and from industry, and is considered by many to be the best cryptographic defense for BGP [7]; indeed, Cisco encouraged and supported the development of the SPV protocol [8]. Using a novel cryptographic mechanism involving constant-time signatures and hash chains, the designers of SPV aim to prevent attacks in which a malicious AS presents forged routing updates to neighboring ASes by illegitimately altering received routing messages. In addition, SPV is more efficient than the secure, but computationally expensive, S-BGP protocol [13].

In this paper we show that it is possible to mount attacks on SPV with high probability. We present simple scenarios in which an adversarial AS receives route announcements corresponding to two or more paths to the same IP prefix and combines this information to forge an illegitimate route advertisement. For example, consider the AS topologies shown in Figure 1, where an AS A advertises a route for an IP prefix to its neighbors; the route eventually propagates to a malicious AS M via two different paths. In these scenarios, our attacks enable M to forge route advertisements for non-existent paths (shown by dotted lines). Such sub-topologies of ASes often appear in practice: our experiments on the AS-level Internet topology from CAIDA's skitter project [6] indicate that 60.4% of all ASes are in a position to mount at least one of the attacks shown in Figure 1.

Indeed, it was part of SPV's design objective to protect BGP from arbitrary route forgeries of this kind. Security against such forgeries is important in the context of BGP. Though routers typically select short AS paths, route selection in BGP is based on complex local policy often unrelated to path length. In fact, local policy is given a higher priority than path length in the BGP decision process [5]; ASes often select routes with longer paths and sometimes select routes based upon intermediate AS numbers in the received paths [10][1]. Furthermore, BGP policies are kept confidential by most ASes, which makes it impossible to reason about when a routing protocol is invulnerable to certain forgeries. In general, it is most desirable that protocols for securing BGP be designed without any a-priori assumptions on the route selection process (as was done in the case of SPV).

VARIANTS OF SPV. SPV's underlying mechanism has two variants: the basic ASPATH protector and the advanced

[1]Modern routers make it easy for ASes to implement such policies. For example, in Cisco routers the sequence of IOS commands `match as-path 11`, `set local-preference 100`, and `ip as-path access-list 11 permit _1234_` results in setting a local preference of 100 for routes containing AS number 1234 anywhere in the ASPATH.



(a) (b) (c) (d)

Figure 1: Example Attack Scenarios: A is the source AS, M is the attacker, and V is the victim AS, to whom M sends forged updates. Dotted lines indicate non-existent route segments that are forged by M in route advertisements it sends to V.

ASPATH protector. The basic ASPATH protector was designed to prevent against a certain class of forgeries but was vulnerable to attacks in situations where the malicious AS has multiple routes to the source (as in our examples above). In order to fix the weaknesses in the basic variant (and make the protocol secure against all possible forgery attacks), the authors of SPV modified it to obtain the advanced ASPATH protector. Our analysis of SPV uncovers subtle flaws in the mechanism used in this modification. We show that the advanced protector, while more complex and less efficient than its basic variant, provides little security improvement. In other words, the entire protocol is as secure as its basic variant.

OTHER CONCERNS. Irrespective of cryptographic weaknesses in SPV, we reconsider several of its design tradeoffs in Section 5. Particularly, we find that constant-time signatures may not be necessary for high performance, and that SPV's underlying attack model may make it unsuitable for real-world deployment.

2. Review

In this section, we review the security needs of BGP and the goals of the SPV protocol.

2.1 Wide-area routing security

The Border Gateway Protocol (BGP) is the sole mechanism for wide-area route dissemination in the Internet. Routing exists on two planes: the data plane—along which packets are forwarded—and the control plane—along which routes are updated. BGP operates on the control plane—routers in different Autonomous Systems (ASes) use it to disseminate routes to each other. The key mechanism used in route dissemination is the BGP Update message, which contains information about an IP prefix and a route, specified as an attribute called ASPATH, via which that prefix is reachable in the network. Each AS, upon receipt of an Update message, can opt to prepend its AS number to the ASPATH of the message and, in turn, send an Update, containing the prepended ASPATH, to its neighboring ASes. Decisions regarding route forwarding (whether to propagate a route to downstream ASes or to filter it) and ranking (choosing the best route among multiple routes to the same prefix) is governed entirely by local policy configurations of ASes.

The problem of securing BGP in the presence of malicious parties presents many challenges, and requires addressing a wide variety of attack scenarios. (See [3] for a survey on this topic.) In this paper, our focus is on a specific class of attacks, namely, those that involve forgery of Update messages.

In BGP, ASes are only allowed to prepend their AS number to the ASPATH attribute of already-received Update messages (or of messages originate by them). *Path forgery* refers to any attack in which a malicious AS, say M, violates this requirement, that is, creates an Update whose ASPATH is not obtained by prepending M to ASPATHs already known to M (and convinces another AS of the validity of such an Update)[2]. Such a violation can adversely affect the normal execution of downstream ASes and, consequently, of the entire protocol. Path forgery can involve *truncation* (the attacker removes some AS numbers from a received ASPATH and propagates the resulting, shortened path), or *modification* (the attacker changes AS numbers on incoming ASPATHs and forwards such modified paths), or both.

2.2 Goals of SPV

SPV is a protocol primarily designed to protect BGP from any path forgeries using efficient symmetric key cryptography [12]. SPV's attack model considers solitary misbehaving ASes that attempt to subvert the normal execution of the protocol; that is, SPV assumes that multiple conspiring ASes cannot mount coordinated attacks. SPV also assumes that the links between any two ASes are private and authenticated; that is, the adversarial AS can neither eavesdrop on nor make modifications to any communication between two other directly-linked ASes.

3. Description of SPV

In this section, we describe the SPV protocol in brief. We highlight those aspects of the protocol which are most relevant to security against path forgery attacks.

3.1 Constant-time signatures

Constant-time signature schemes are fundamental to the design of SPV. A constant-time signature scheme is a weaker version of a standard public-key signature scheme in which

[2]Hu *et al.* [12] use the term *falsification* instead of *forgery*.

security is guaranteed against an adversary who can acquire the signatures of only a small (constant) number of messages. For example, in a one-time signature scheme, it is hard to forge signatures given only a single example message-signature pair, but the scheme may be insecure given more such pairs. In general, an m-time signature scheme (for some constant m) resists forgeries provided that no more than m signatures for a given key are known by the adversary. Such signature schemes typically admit more efficient implementations than ordinary public-key signatures.

3.1.1 M-HORS m-time signatures

SPV uses a modified version of the Hash to Obtain Random Subsets (HORS) m-time signature scheme [21], which in turn is an improvement of the BiBa signature scheme [20]. We refer to this modified scheme as M-HORS (Modified HORS) and describe it below. (Figure 2(a) illustrates the process.)

As with any signature scheme, M-HORS has both public and private keys, where the private keys are used for signing and the public keys are used for verifying signatures. In M-HORS, key generation involves choosing a key K at random and generating N values, $v_0, \ldots, v_{N-1}$, with each $v_i = F_K(i)$, where F is a block cipher such as AES and N is a security parameter. K serves as the secret key of the signature scheme and the v_i's are called *private values*.

Private values are hashed using a hash function H (such as SHA-1) into N values $u_0, \ldots, u_{N-1}$, referred to as *public values*: each u_i is equal to $H(v_i)$. Finally, a hash tree is built, using H, over the public values (that is, with these values as leaves). The root of this hash tree, R, is the public key.

To sign a message M, the signer (who knows the secret key and, thus, all the private values) picks a set $S = \{s_1, s_2, \ldots, s_n\}$ of n numbers in the range 0 to $N-1$, based on the value $H(M)$; n is another security parameter. The signature of M consists of two parts: the set of private values corresponding to set S (the set $v_S = \{v_i\}_{i \in S}$) and the smallest set U of values in the hash tree required to verify v_S (to compute R given v_S). To verify a signature on a message M (which consists of private values v_S and hash tree values U), the verifier re-computes the hash tree based on the values v_S and U and then compares the root to the public key R[3].

Figure 2(a) illustrates this with an example where $N = 4$ and $n = 2$. Suppose that a message M hashes to a set $S = \{0, 2\}$. The signature of M consists of the set of private values $v_S = \{v_0, v_2\}$ (black dots) and the values $U = \{u_1, u_3\}$ of the hash tree (gray dots). To verify the signature, one re-computes the hash tree and checks that the root matches the public key R.

3.2 SPV setup

As part of setup in the SPV protocol, every AS A constructs a sequence of secret values $s_1, s_2, \ldots, s_l$ (where l equals the length of the longest possible AS-level path that any route announcement will traverse) using a one-way hash function H_1. That is, A picks s_1 at random and computes $s_{i+1} = H_1(s_i)$ for $i = 1, \ldots, l-1$. Each s_i is treated as a secret key of the M-HORS scheme of Section 3.1. Let p_i denote the public key corresponding to the secret key s_i. A also builds a hash tree over these public keys and the root of this tree is labeled R (this node is used in the verification process). The entire construction is called the *basic* ASPATH *protector*; an example with $l = 4$ is shown in Figure 2(b). In SPV, a single function H is used for all hash operations; so $H_1 \equiv H$. (We choose to use a special notation for H_1 for reasons that will become clear in Sect. 4.) SPV suggests to instantiate H with the Matyas, Meyer, and Oseas hash function construction [16] using the AES block cipher.

3.3 Basic route forwarding

We first present a simplified version of SPV's route forwarding. Update messages in SPV include, besides the usual attributes of BGP Updates, signatures created using the ASPATH protector. Suppose that A wants to send an announcement for a prefix to a neighbor B. A signs the path $\langle \mathsf{B}, \mathsf{A} \rangle$ using the secret key s_1 to get a signature σ_1. Then A sends an Update message having ASPATH $= \langle \mathsf{A} \rangle$, tagged with σ_1 and the secret key s_2, to B.

On receipt of this message, B validates it by re-computing the whole ASPATH protector. Namely, it first computes the secret keys $s_3, \ldots, s_l$ from s_2, uses them to compute the corresponding public keys and then re-computes the root public key using the public keys $p_1, \ldots, p_l$ (and multiple applications of H). Verification consists of comparing the re-computed root public key with the original root public key R, which is assumed to be known to all ASes[4].

After validating the message, if B decides to forward information about the path $\langle \mathsf{B}, \mathsf{A} \rangle$ to a neighbor, say C (based on its local policies), it first signs the ASPATH $\langle \mathsf{C}, \mathsf{B}, \mathsf{A} \rangle$ under secret key s_2 (using M-HORS), thus creating a signature σ_2. The Update message from B to C consists of ASPATH = $\langle \mathsf{B}, \mathsf{A} \rangle$, tagged with σ_1, σ_2 and s_3. (Figure 2(b) shows these values; we use gray dots to represent the signatures σ_1, σ_2 and a black dot for the secret value s_3.) C validates the message in the same way as B, while also ensuring that the path is loop-free and that B is the last AS in the ASPATH. Further Update messages are created and propagated similarly.

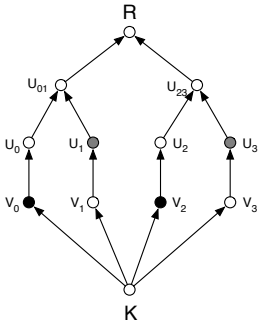
If routes to A are propagated along another path (say, A sends an update to some AS B′ and B′ forwards it to another AS C′, and so on), the *same* chain of keys $s_1, \ldots, s_l$ is used for tagging the corresponding Updates, which means that a secret key s_i could be known to multiple ASes in the network (in our example, B and B′ both possess $s_2, s_3, \cdots$ and C and C′ both possess $s_3, s_4, \cdots$). This sharing of keys is an undesirable feature of SPV, and, in fact, it forms the basis of all our attacks.

3.4 Postmodification

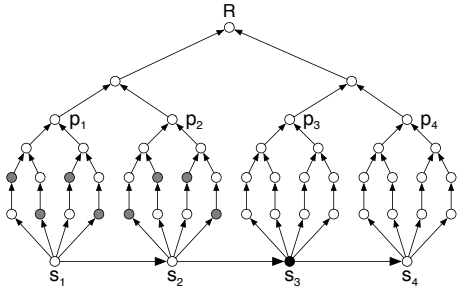
The full SPV protocol is more involved than the basic mechanism presented above. In order to counter forgeries that could

[3] In the original HORS scheme [21], the secret key consists of all private values and the public key of all public values; no hash tree or PRF applications are involved. Security is proven under the assumption that H satisfies the "subset-resilience" property, introduced and defined in [21].

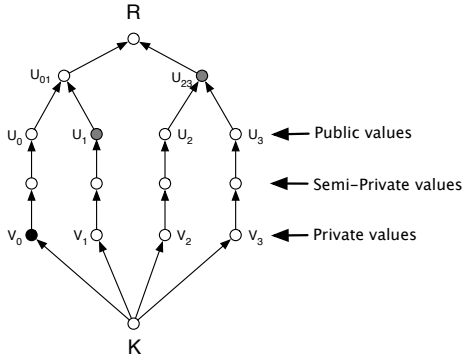
[4] In fact, R serves as a short-term public key or an "epoch" public key and multiple such public keys are authenticated using a "multi-epoch" public key. See the original paper for details [12].



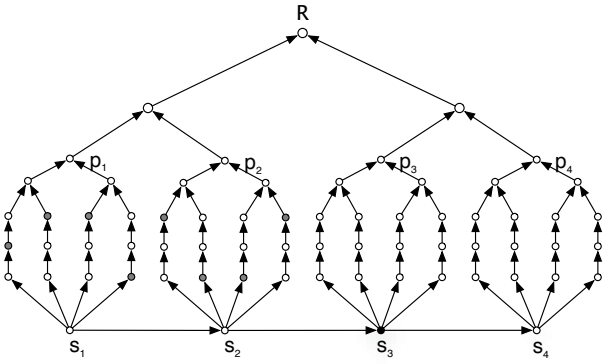
(a) A M-HORS signature



(b) The basic ASPATH protector



(c) M-HORS with postmodification



(d) The advanced ASPATH protector

Figure 2: The use of constant-time signatures in SPV.

arise due to the sharing of keys between ASes, the authors of SPV modify the basic ASPATH protector (Fig. 2(b)) into what is called the *advanced* ASPATH protector (Fig. 2(d)). The advanced protector uses a slight variant of the M-HORS scheme: each public value u_i is obtained by hashing the corresponding private value v_i *twice* rather than once, as shown in Figure 2(c); that is, $u_i = H(H(v_i))$. The intermediate hash value, $H(v_i)$, is referred to as a *semi-private value*. Signatures are created as in basic forwarding, except that now any party, given the signature on a path, can modify it by "degrading" some of the n revealed private values to semi-private values (by applying H once on them). The degraded signature can still be verified but since the function H is assumed to be one-way, no adversary can recover the original signature.

To understand how "degrading" is used in SPV, consider again the three-AS example from the previous subsection, as shown in Figure 2(d). When A wishes to send an Update to B, it sends the same message, tagged with σ_1 and s_2, as before. However, before forwarding the announcement to C, B modifies σ_1 by degrading one of the private values contained in it to get a new signature, say $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$; this is called *postmodification* of σ_1. The private value to degrade is determined based on the hash of $\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle$, $H(\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle])$. The motivation for postmodification is to prevent a downstream AS, say D, from truncating a valid route of the form $\langle \mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A}\rangle$ to $\langle \mathsf{D},\mathsf{B},\mathsf{A}\rangle$ even when it knows s_2 (via some other path), since this would most likely require recovering σ_1 from $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$, which, essentially, means inverting the hash function on a semi-private value. When C forwards the information about the route $\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle$ to D, it may degrade another value in $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ to get a signature $\sigma_1^{\langle \mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ as well as some value in σ_2 to get $\sigma_2^{\langle \mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A}\rangle}$.

Hu *et al.* don't lay down any strict rules for deciding *which* private value should be degraded based on a given hash. In our attacks, we use the following natural convention: if a signature σ contains k private values (and $n-k$ semi-private values) and a private value needs to be degraded in σ based on the hash $h := H(p)$ (p being an ASPATH), then the first $\lceil \log_2(k) \rceil$ bits of h are mapped to a number $i \in \{1, 2, \cdots, k\}$ and the ith private value is chosen for degrading. In the sequel, we use the notation $[h]_k$ to denote the number i derived from the hash h in this manner. We remark that our attacks are independent of the

convention used for degrading; any other convention would result in the same (or possibly worse) attack probabilities.

3.5 Concrete parameters

SPV uses an instantiation of M-HORS (with postmodification) such that it is 15-time secure. This is justified since it is unlikely that any AS in the Internet will receive more than 15 Updates for the same route. 15-time security is achieved by setting N (number of private values) to 256 and n (number of private values revealed per signature) to 6. The value of l (length of the ASPATH protector) is chosen to be 14. Numerous parameters, denoted $\mu_0 \geq \mu_1 \geq \cdots \geq \mu_{14}$, govern how private values are degraded while routing: when a signature σ has traversed i hops, it contains μ_i private values and $n - \mu_i$ semi-private values. (In our example, the Update that C sends to D contains a signature $\sigma_1^{\langle \mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A}\rangle}$, that has μ_2 private values, and a signature $\sigma_2^{\langle \mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A}\rangle}$, that has μ_1 private values.) In general, any signature σ_i^p contains $\mu_{|p|-i-1}$ private values. SPV sets $\mu_0 = 6$, $\mu_1 = \mu_2 = 5$ and $\mu_3 = 4$. (The rest are not relevant to the attacks).

4. Attacks

Next we present our attacks on SPV, all of which use the same general approach: a malicious AS M receives Update messages corresponding to two paths, p_1 and p_2, originated by AS A, with path lengths l_1 and l_2 respectively such that $l_2 > l_1$. M then uses the secret key s_{l_1+1} (obtained from the Update message for p_1) to "sign in" a suitably-chosen AS into the last position of p_2. The AS number to forge is selected in a manner such that the signatures received along p_2 can be used to authenticate the forged path to the victim AS. (Particularly, this can be done even without inverting the hash function H or breaking M-HORS.) Using this approach, M can perform a variety of modification or truncation attacks on p_2.

We refer to such attacks as *multi-path forgeries*. As we will discuss, in most scenarios, the longer the path p_2 is than p_1, the greater the chance of attack success. Also, if more than one path longer than p_1 is available, the probability of attacking all paths increases.

Figures 1 (a) and (b) show example attack scenarios in which the forgery is a multi-path modification; these forgeries are successful with high probability if M has a sufficient number of choices for the AS number D that it wants to forge. Figures 1 (c) and (d) show scenarios in which multi-path truncation can be carried out; these can occur with probability $1/6$ and $1/5$ respectively.

It is important to note that our attacks succeed against SPV even in the presence of postmodification. SPV uses postmodification with the sole objective of countering multi-path forgeries; our findings, however, establish that the protocol is still weak against these attacks.

4.1 Multi-path modification attacks

Consider the scenario shown in Figure 1(a). Suppose M receives Updates corresponding to both routes to A: $\langle \mathsf{M},\mathsf{A}\rangle$ and $\langle \mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle$. The Update for $\langle \mathsf{M},\mathsf{A}\rangle$ contains the secret key s_2; that for $\langle \mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle$ contains a signature $\sigma_1^{\langle \mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ on the message $\langle \mathsf{B},\mathsf{A}\rangle$. (This signature is, in fact, the same as $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ since SPV's parameters dictate that that μ_1 and μ_2 be equal.) Combining these two pieces of information, M can send to the victim V a valid (yet forged) Update for a path of the form $\langle \mathsf{M},\mathsf{D},\mathsf{B},\mathsf{A}\rangle$. First, M selects the AS number D in a manner such that $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ can be used to obtain another postmodified version of σ_1, namely $\sigma_1^{\langle \mathsf{V},\mathsf{M},\mathsf{D},\mathsf{B},\mathsf{A}\rangle}$. In other words, it picks D such that the private value degraded while modifying σ_1 to $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ (based on $h_1 := [H(\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle)]_n$) is the same as one of the two values that would be degraded while modifying σ_1 to $\sigma_1^{\langle \mathsf{V},\mathsf{M},\mathsf{D},\mathsf{B},\mathsf{A}\rangle}$ (based on $h_2 := [H(\langle \mathsf{D},\mathsf{B},\mathsf{A}\rangle)]_n$ and $h_2' := [H(\langle \mathsf{V},\mathsf{M},\mathsf{D},\mathsf{B},\mathsf{A}\rangle)]_{n-1}$). Thus, a good choice for D is one that ensures that either of the hash values h_2 or h_2' equals h_1. Once an AS number D with this property has been chosen, M can modify $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$ (by suitably degrading one of the private values in it) to get $\sigma_1^{\langle \mathsf{V},\mathsf{M},\mathsf{D},\mathsf{B},\mathsf{A}\rangle}$. Then, M uses the secret key s_2 (received via $\langle \mathsf{M},\mathsf{A}\rangle$) to forge the remaining signatures for $\langle \mathsf{M},\mathsf{D},\mathsf{B},\mathsf{A}\rangle$.

How many choices of D are good for the attack to work? If the hash function is perfectly random (which is, in fact, the hardest scenario for our attacks), the probability that h_1 equals either h_2 or h_2' for any fixed value of D is exactly $2/n = 2/6 = 1/3$. This probability is obtained using a slightly stronger assumption, namely that the map from ASPATHs to elements in $\{1, \cdots, n\}$ used in SPV is perfectly random. This means that out of all possible choices for D, say x, about $x/3$ would be effective in the attack.

Path lengthening attacks can be mounted similarly. Consider the scenario shown in Figure 1(b) and suppose that M wishes to lengthen the path $\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle$ to advertise a new (forged) path $\langle \mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle$. If M receives Update messages from both $\langle \mathsf{M},\mathsf{A}\rangle$ and $\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle$, this can be achieved quite easily. The former contains the secret key s_2 (from which other keys $s_3, s_4, \ldots$ can be computed) and the latter contains the signature $\sigma_1^{\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle}$ on $\langle \mathsf{B},\mathsf{A}\rangle$ which is postmodified by B according to $h_1 := [H(\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle)]_n$. The Update message for $\langle \mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle$ sent to V should contain a signature $\sigma_1^{\langle \mathsf{V},\mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle}$, postmodified according to $h_2 := [H(\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle)]_n$ and to $h_2' := [H(\langle \mathsf{V},\mathsf{M},\mathsf{C},\mathsf{B},\mathsf{A}\rangle)]_{n-1}$. As in the previous attack, M can create such a signature if C is selected to ensure $h_1 \in \{h_2, h_2'\}$, which occurs with probability at least $1/3$. So, if M has x different choices for the AS number C, then about $x/3$ of them work for the attack.

4.2 Multi-path truncation attacks

We now show the feasibility of multi-path truncation attacks against SPV. Consider the scenario in Figure 1(c) (the same as that in Figure 1(a)). We argue that in this setting, M can convince V of the ASPATH $\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle$ (which doesn't exist in the network) with probability $1/6$. M's aim is to be able to forge $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle}$—the postmodification of A's signature on $\langle \mathsf{B},\mathsf{A}\rangle$, σ_1—as a different postmodification $\sigma_1^{\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle}$ of the same signature. A sufficient condition for this is that $H(\langle \mathsf{M},\mathsf{B},\mathsf{A}\rangle)$ selects the same value to degrade in σ_1 as does $H(\langle \mathsf{C},\mathsf{B},\mathsf{A}\rangle)$. The probability that this condition is true is at least $1/n = 1/6$. (It is *exactly* $1/n$ if the map from ASPATHs to the set $\{1, \cdots, n\}$ is perfectly random.) This means that

of all possible subgraphs of the AS-level Internet topology of the kind illustrated in Figure 1(c), at least $1/6$ will succumb to this attack.

If the difference between the lengths of the paths by which M is linked to A is greater than 2, the chance of success is greater. Figure 1(d) shows such a scenario—two paths connecting M to A are $\langle \mathsf{M}, \mathsf{A} \rangle$ and $\langle \mathsf{M}, \mathsf{D}, \mathsf{C}, \mathsf{B}, \mathsf{A} \rangle$. The message from A contains s_2 as before while that from D contains $\sigma_1^{\langle \mathsf{M},\mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A} \rangle}$. The hope now is that the value $[H(\langle \mathsf{M}, \mathsf{D}, \mathsf{C}, \mathsf{B}, \mathsf{A} \rangle)]_5$ (which determines how $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A} \rangle}$ is postmodified to $\sigma_1^{\langle \mathsf{M},\mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A} \rangle}$) equals $[H(\langle \mathsf{V}, \mathsf{M}, \mathsf{C}, \mathsf{B}, \mathsf{A} \rangle)]_5$. This happens with probability $1/5$ since there are 5 private values in $\sigma_1^{\langle \mathsf{C},\mathsf{B},\mathsf{A} \rangle}$ (and one is degraded to get $\sigma_1^{\langle \mathsf{M},\mathsf{D},\mathsf{C},\mathsf{B},\mathsf{A} \rangle}$).

In the evaluation of postmodification with respect to multi-path truncation attacks, Hu *et al.* present several attack probabilities for particular scenarios in the Internet's AS-level topology [12]; to our knowledge, they do not consider the attack scenarios we present. Our attacks show that postmodification can mitigate the risk of multi-path forgeries only to a very small extent, by reducing attack probabilities by a small constant factor.

4.3 Topological analysis

To understand the feasibility of our attacks, we performed several simulations to determine how often subgraphs such as those in Figure 1 appear in the Internet's AS connectivity graph. To ensure that the AS graph we considered reflected real Internet paths, we opted to use AS connectivity extracted from CAIDA's skitter project [6]; this data reflects actual paths rather than advertised routes. We treated each AS as a node and each visible AS connection as an edge in a graph; we performed a depth-limited breadth-first search from each node and counted the number of times that one of our attack scenarios appeared as a subgraph. From this, we determined that 60.4% of all ASes are in a position to mount at least one of the attacks shown in Figure 1.

4.4 Attack summary

We stress that the susceptibility of SPV to our attacks is not based on weaknesses, if any, in the underlying cryptographic primitives, but rather in the manner in which these primitives are composed in SPV. For example, our attacks never use the fact that the M-HORS scheme in use is 15-time secure rather than secure in the standard sense of digital signatures; in fact, the attacks never involve forging a signature for an unknown key.

Our techniques can be easily generalized to a scenario in which the malicious AS M receives Updates for several (more than two) paths to A (of various lengths) and its goal is to forge Updates by illegitimately modifying *some* received path. The techniques are not applicable in the specific case wherein M wants to attack (that is, modify or truncate) only the single shortest received path; SPV may be secure against such attacks. However, rigorously arguing about SPV's security even against such specific attacks is difficult. For example, hash chains are used in the ASPATH protector in a non-standard manner: each secret value s_i in the hash chain is used not only as an input to a hash function but also for signing messages (in M-HORS), which could affect the security of both the hash chain and the signature scheme. It is commonly accepted that a single key should not be used as input to multiple cryptographic operations. In principle, it is possible to instantiate the hash function H_1 used in the hash chain and M-HORS in a way such that the former is one-way and collision-resistant, the latter is a secure m-time signature scheme and still, an attacker who is given only $H_1(s_j)$, is able to successfully forge signatures under s_j.

While it is possible to fix this particular weakness in SPV's design quite easily[5], such a fix would still not recover security against our forgery attacks. Modifying SPV's concrete parameters can do little to improve its security, but can significantly decrease efficiency: we could increase the value of n to reduce the success probability in our attacks, but that would require raising N to maintain security of M-HORS. Thus, we believe that the design of the ASPATH protector (with or without postmodification) is unsuitable to prevent against path forgeries, and must be replaced if strong security against forgery attacks is desired.

5. Discussion

Thus far we have detailed several specific weaknesses of SPV's ASPATH protector. While these alone indicate that SPV is unsafe for deployment, we also consider a more holistic view of SPV next, and reconsider several of SPV's design decisions.

5.1 Reconsidering constant-time signatures

In the past, protocols such as S-BGP used asymmetric signatures schemes to provide strong unforgeability; unfortunately, this comes at the price of high computational overhead. Motivated by this bottleneck, Hu *et al.* use constant-time signatures in SPV due to their significantly better performance. Irrespective of their security, M-HORS and other constant-time signature schemes trade off communication overhead for this speedup. This was a known consequence to SPV's designers; SPV incurs a network bandwidth overhead that is a factor 2.731 greater than that of S-BGP [12].

In practice, it may be acceptable to trade off bandwidth for performance. Unfortunately, another hidden trade-off of using such signature schemes is their implementation complexity. Unlike several asymmetric signature schemes that are self contained and standardized, M-HORS is not standard, and its use in SPV is integral to the ASPATH protector.

Furthermore, the parameters for M-HORS used in SPV trade off security for space and performance, with natural con-

[5]One possible implementation would be to replace the hash chain with keys generated using a *forward-secure pseudorandom generator (FS-PRG)* [1]. This would simultaneously guarantee the "one-wayness" property required by the application and preserve the pseudorandomness of the keys (which is needed for M-HORS to be secure). Krawczyk's approach to building *forward-secure signatures (FSS)* [14] uses the same ideas; indeed, it might be possible to exploit specific constructions of FSS schemes to get more secure designs for the ASPATH protector.

sequences: (i) brute-force forgeries in SPV can be carried out with probability as high as 2^{-22} by any adversary (who is given 15 signatures under the same signing key)[6]; and (ii) due to limits on the size of BGP Update messages, at most 14 M-HORS signatures (using the suggested parameters) can be communicated in any single message. That is, SPV can only authenticate routes with less than 14 hops. While this latter constraint is minor—few Internet paths are longer than 14 hops—its consequence in the design of SPV is more important: SPV must, by its construction, compute a hash chain and corresponding M-HORS signatures for all 14 hops (even if the path in question is only, say, 2 hops) because the public key must be computed over the longest possible ASPATH protector.

As a result of this bandwidth increase, it is natural to consider signature aggregation schemes, since it has been shown [25] that using such techniques, the communication overhead of S-BGP can be reduced to almost *constant* (as opposed to linear) in the length of the ASPATH being signed. Unfortunately, unlike those proposed for RSA-based signature schemes [15], there are no known signature aggregation techniques for constant-time signature schemes.

Instead of attempting to remedy this situation by tweaking M-HORS or the ASPATH protector, we believe that protocol designers should reconsider the use of constant-time signatures. One of the primary motivations for the use of M-HORS in SPV was its performance. However, we note that there are signature schemes that are secure in the standard sense and are simultaneously more efficient than those most commonly used. One candidate is the ESIGN scheme of Okamoto *et al.* [11, 19]. ESIGN is a conventional signature scheme that is significantly faster than and has comparable security to RSA—a study conducted as part of the CRYPTREC project by Menezes *et al.* found that ESIGN and RSA provide the same security at equal key lengths [17]—though it does not double as a public-key encryption scheme as RSA does.[7] A preliminary experimental analysis indicates that using ESIGN, it may be possible to achieve efficiency better than that of the instantiation of M-HORS used in SPV while also assuring much better security. In Table 1, we consider the efficiency of 15-time secure M-HORS as used in SPV and conventional digital signature schemes. We obtained these benchmarks using a standard cryptographic library, Crypto++ 5.2.1 [9], on a Pentium 4 2.8 GHz machine using GCC 4.0 with standard compiler optimizations (-O2) enabled. We briefly consider how to use ESIGN to build a fast BGP security protocol in Section 6.

Operation	Time
AES (per block)	0.287 μsec
M-HORS (1278 AES calls)	366 μsec
1024-bit ESIGN (sign/verify)	469 μsec/175 μsec
2048-bit ESIGN (sign/verify)	1131 μsec/457 μsec
1024-bit RSA (sign/verify)	6172 μsec/185 μsec
2048-bit RSA (sign/verify)	34482 μsec/472 μsec

Table 1: Comparison of M-HORS and standard digital signatures. In SPV, M-HORS is always called once for each hop in the ASPATH protector, which is of a fixed size; the default is 14. ESIGN, in contrast, would only need to be called once per actual hop in the ASPATH, which is usually much less than 10.

5.2 Collusion and Eavesdropping Attacks

SPV's attack model assumes that multiple malicious ASes do not collude with each other in an attempt to perform forgeries. While this may hold for some subset of deployments, this assumption does not correspond well with practical threats to BGP. For example, there are several instances in which numerous ASes are owned and operated by a single organization and such ASes can trivially mount coordinated attacks on the protocol. (Even if such ASes were not themselves malicious, if their routers were compromised, an attacker could wield substantial power.) By colluding, ASes can share keys, signatures, and any topological information that they know. Such an exchange is potentially useful to forge routes that are completely non-existent in the network.

Collusion attacks are simple to mount against SPV. For example, it is possible for two malicious ASes M_1 and M_2 (that are colluding with one other) to introduce (or "sandwich") arbitrary AS numbers *between* themselves and to propagate ASPATHs of the form $\langle \mathsf{M}_1, \mathsf{X}, \mathsf{M}_2, * \rangle$ even when X is not connected to either M_1 or M_2. Note that such "sandwiching" attacks do not work against S-BGP [13], soBGP [24], or psBGP [23], which require that every AS sign Updates using its own secret key (and not a secret key sent to it by a neighbor).

While such collusion attacks were expected to be possible by SPV's designers, another assumption in SPV's model is more insidious: its reliance on encrypted, authenticated AS-to-AS channels. Clearly, if SPV is implemented without secure AS-to-AS links, any eavesdropper can use the secret keys sent between ASes to its advantage and forge almost arbitrary ASPATHs. Combined with a TCP hijacking attack [2], a malicious entity eavesdropping on a link (A, B) could even convince another AS X of the validity of the ASPATH $\langle \mathsf{X}, \mathsf{Y}, \mathsf{B}, \mathsf{A} \rangle$ where Y is a neighbor of X, even though there is no such path—again, such attacks are not possible against other protocols [13, 23, 24] because they do not send secret keys between ASes. Hu *et al.* [12] suggest that a protocol like IPSec be used to secure AS-to-AS channels. This requirement seems reasonable, but we note that it too has a subtle consequence: AS-to-AS authentication and key distribution are required. An important design goal of SPV was to simplify key management. Unfortunately, as a result of IPsec, ASes must perform pairwise authentication with each neighbor and establish a secret key, and thus, for tier 1 ISPs,

[6]This forgery probability presented in Section 5.1.1 of the original SPV paper [12] is based on the assumption that the hash function used in M-HORS behaves as a perfectly random function. This assumption is stronger than that used to prove security of M-HORS in the original paper of Reyzin *et al.* [21], where the hash function is assumed to satisfy only subset-resilience (a weaker property than that of being a perfectly random function)

[7]ESIGN was patented in 1986 by NTT; however, this patent has recently expired and the scheme is now unencumbered.

this would require thousands of security associations. Alternatively, they could have simply received authenticated public keys for thousands of other ASes and used S-BGP.

6. Future directions

In this paper, we have presented a security analysis of the SPV protocol [12] for securing BGP, and have uncovered attacks that we believe should be guarded against before SPV is considered for deployment. In light of our analysis of SPV and its use of constant-time signatures, we believe that a simple step can be made to improve SPV, albeit in its current restricted security model. Particularly, we can modify an alternative protocol suggested by Hu *et al.* (that combined elements of SPV and S-BGP to increase the deployability of S-BGP [12]) as follows. Instead of using a hash chain, we can use a certificate chain of digital signatures that is generated on-the-fly to provide route authenticity; each AS generates an asymmetric keypair for the next hop and certifies the public key. Since these keypairs are relatively short lived, 1024-bit ESIGN is more than sufficient; 2048-bit ESIGN can be used for longer lived prefix keys. Such a protocol would have superior security and performance to SPV.

While it may be possible patch SPV in this manner, we believe routing security must be formalized using appropriate cryptographic definitions to enable analysis of the security of existing protocols, including SPV. Despite the extensive literature on securing BGP, there seems to have been little work on this subject (though Buttyán *et al.* have taken a step toward the provable security of ad-hoc wireless routing protocols [4]). This gap may exist due to the inherent difficulty of modeling BGP security accurately. For example, cooperation between multiple malicious ASes cannot be neglected: in practice, since some organizations possess multiple AS numbers, collusion between ASes is trivial. Furthermore, ASes cannot be treated as atomic routing entities, as most ASes are composed of many routers, each advertising different BGP routes to neighbors. While these are only examples of complicating factors, they indicate that developing an adequate model of routing security is a large task in itself, though such a model is necessary if we are to have confidence in secure routing protocols in the future.

7. References

[1] M. Bellare and B. Yee. Forward security in private key cryptography. In *Proceedings of CT-RSA*, Apr. 2003.

[2] S. M. Bellovin. Security problems in the tcp/ip protocol suite. *Computer Communications Review*, 2(19):32–48, 1989.

[3] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of BGP security: Issues and solutions. Technical Report TD-5UGJ33, ATT Research, Apr. 2005.

[4] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of ACM SASN*, Oct. 2004.

[5] M. Caesar and J. Rexford. BGP routing policies in ISP networks. Technical Report UCB/CSD-05-1377, University of California, Berkeley, Mar. 2005.

[6] CAIDA Skitter Project. `http://www.caida.org/tools/measurement/skitter/`.

[7] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocols. In *Proceedings of ACM SIGCOMM*, Sept. 2006.

[8] Cisco Systems. Personal communication, Apr. 2006.

[9] Crypto++ library. `http://www.eskimo.com/~weidai/cryptlib.html`.

[10] N. Feamster. Personal communication, Oct. 2005.

[11] A. Fujioka, T. Okamoto, and S. Miyaguchi. ESIGN: An efficient digital signature implementation for smart cards. In *Proceedings of EUROCRYPT*, Apr. 1991.

[12] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: secure path vector routing for securing BGP. In *Proceedings of ACM SIGCOMM*, Sept. 2004.

[13] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4), 2000.

[14] H. Krawczyk. Simple forward-secure signatures from any signature scheme. In *Proceedings of ACM CCS*, Nov. 2000.

[15] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Proceedings of EUROCRYPT*, May 2004.

[16] S. Matyas, C. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithms. IBM Technical Disclosure Bulletin 27:5658-5659, 1985.

[17] A. Menezes, M. Qu, D. Stinson, and Y. Wang. Evaluation of security level of cryptography: Esign signature scheme. CRYPTREC Project, Japan, Jan. 2001.

[18] O. Nordström and C. Dovrolis. Beware of bgp attacks. *SIGCOMM CCR*, 34(2), 2004.

[19] T. Okamoto and J. Stern. Almost uniform density of power residues and the provable security of ESIGN. In *Proceedings of ASIACRYPT*, Nov. 2003.

[20] A. Perrig. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of ACM CCS*, Nov. 2001.

[21] L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Proceedings of ACSIP*, July 2002.

[22] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and whisper: Security mechanisms for BGP. In *Proceedings of USENIX/ACM NSDI*, Mar. 2004.

[23] T. Wan, E. Kranakis, and P. van Oorschot. Pretty secure BGP (psBGP). In *Proceedings of ISOC NDSS*, Feb. 2005.

[24] R. White. Securing BGP through Secure Origin BGP (soBGP). The Internet Protocol Journal, Sept. 2003.

[25] M. Zhao, S. W. Smith, and D. M. Nicol. Aggregated Path Authentication for Efficient BGP Security. In *Proceedings of ACM CCS*, Nov. 2005.