An Unknown Key-Share Attack on the MQV Key Agreement Protocol

BURTON S. KALISKI, JR.
RSA Laboratories

The MQV key agreement protocol, a technique included in recent standards, is shown in its basic form to be vulnerable to an unknown key-share attack. Although the attack's practical impact on security is minimal—a key confirmation step easily prevents it—the attack is noteworthy in the principles it illustrates about protocol design. First, minor "efficiency improvements" can significantly alter the security properties of a protocol. Second, protocol analysis must consider potential interactions with all parties, not just those that are normally online. Finally, attacks must be assessed in terms of system requirements, not just in isolation.

Categories and Subject Descriptors: E.3 [**Data**]: Data Encryption—*public key cryptosystems*, *standards*

General Terms: Algorithms, security

Additional Key Words and Phrases: Key agreement, MQV, protocol design, unknown key-share attack

1. INTRODUCTION

The design of key establishment protocols continues to be one of the most challenging areas of security research. Diffie and Hellman [1976a] observed the importance of such protocols in their seminal work motivating the definition of public-key cryptography. Shortly thereafter they gave the first concrete protocol [Diffie and Hellman 1976b] for *key agreement* in which two parties establish a secret key to which both have contributed key material without having previously shared a secret. Since then, many other key agreement protocols have been proposed. One of the most recent is the MQV protocol, first published in 1995 by Menezes et al. [1995b] and subsequently revised in 1998 with Law and Solinas [Law et al. 1998].

Author's address: RSA Laboratories, 20 Crosby Drive, Bedford, MA 01730; email: bkaliski@rsasecurity.com.

Two common attributes desired for key agreement protocols are:

—(*implicit*) *key authentication*: an agreed-upon secret key should be known (or knowable) only by identified parties;

—*forward secrecy*: an agreed-upon secret key should remain secret, even if both parties' long-term key material is compromised.

Another desirable attribute is resistance to *unknown key-share attacks*. In an unknown key-share attack, an opponent, say Eve, coerces honest parties Alice and Bob into establishing a secret key where at least one of Alice and Bob does not know that the secret key is shared with the other. For example, Eve may coerce Bob into believing he shares the key with Eve, while he actually shares the key with Alice. The "key share" with Alice is thus unknown to Bob.

Unknown key-share attacks on key agreement protocols were first discussed in Diffie et al. [1992] and have been found in a number of protocols including MTI/A0 [Matsumoto et al. 1986] (see Menezes et al. [1995c, 1995b]) and the STS-MAC variant of the Station-to-Station (STS) protocol [Diffie et al. 1992] (see Blake-Wilson and Menezes [1999]). Resistance to unknown key-share attacks was one of the attributes claimed of both the 1995 version of the MQV protocol (a correct statement; see Section 5.1) and the 1998 version. Unknown key-share attacks are also referred to as *source-substitution attacks* (see Menezes et al. [1997, Note 12.54]) since the opponent coerces a party into believing that a message (or key) is from a different source than the actual source. A good overview of unknown key-share attacks can be found in Baek and Kim [2000].

Since the opponent in an unknown key-share attack does not obtain the secret key, the opponent cannot decrypt or modify messages exchanged by the parties. However, the opponent may be able to take advantage of the parties' false assumptions about who shares the key. For instance, Alice may assume that if she integrity-protects a message with the secret key and sends the result to Bob, then Bob will assume the message is from Alice. But Bob will instead assume the message is from Eve. Two classic exploitations of these false assumptions are the following.

—Bob is a bank and Alice sends him an anonymous digital coin, encrypted with the shared secret key, for deposit into her account. Believing that the key is shared with Eve, and based on the fact that the digital coin is self-authenticating, Bob assumes the coin is from Eve and deposits it into Eve's account instead. (See Diffie et al. [1992].)

—Bob is a manager and sends Eve an order (like the apt, "You're fired!"), integrity-protected with the shared secret key. Believing that the key is shared with Bob, Alice assumes the order is for her and follows it.

Prudent protocol engineering (see Abadi and Needham [1996] and Anderson and Needham [1995]) can deal with these errors, for instance by requiring that messages identify the sender and recipient or by including a key confirmation step by which the intended participants can verify one another's possession of the secret key. Indeed, the authors of the MQV protocol strongly recommend

key confirmation. Nevertheless, the existence of an unknown key-share attack on a protocol that was intended to resist it provides another example of the challenges in protocol design.

2. THE MQV PROTOCOL

The MQV protocol [Law et al. 1998, IEEE 2000] is a key agreement protocol in the Diffie–Hellman family, providing key authentication and forward secrecy through a combination of static and ephemeral key pairs. The protocol may be defined over any finite commutative group G with a hard discrete logarithm problem, for instance, a multiplicative group or an elliptic curve group over a finite field.

Domain parameters in the MQV protocol include:

—P, the generator, an element of large prime order in the group G;

—n, the order of the element P;

—h, the cofactor, defined as $h = |G|/n$, where $|G|$ is the order of the group G (for technical reasons it is required that $\gcd(n, h) = 1$); and

—μ, a mapping from group elements (other than the identity element O) to integers in the range $[2^m, 2^{m+1} - 1]$, where m is the bit length of $\sqrt{n}$.[1]

(There may be additional restrictions on the group depending on the implementation, but these are not relevant to the attack presented here.)

In the following, group operations are written additively (i.e., $A + B$ or cA), consistent with notation for elliptic curve cryptography, where the MQV protocol is typically considered [ANSI 2001]. The operation cA is a *scalar multiplication*; the multiplier c is typically an integer between 1 and $n - 1$. The group identity element is denoted O.

A key pair in the protocol consists of a public key T, which is a group element, and a private key t, which is an integer in the range $[1, n-1]$, such that $T = tP$.

As input to the protocol, Alice has a static key pair (W_A, w_A) and Bob has a static key pair (W_B, w_B), where each key pair is certified and conveyed in a certificate and is presumed to be known and verified by the other party. A static key pair will typically be valid for multiple runs of the protocol.

The basic protocol of interest here, called Protocol 1 in Law et al. [1998], consists of the following steps conducted by two parties Alice and Bob. We call this protocol the "basic protocol" to distinguish it from other protocols in Law et al. [1998] and previous versions.

1. Alice generates an ephemeral key pair (R_A, r_A). She sends the ephemeral public key R_A to Bob.
2. Bob generates an ephemeral key pair (R_B, r_B). He sends the ephemeral public key R_B to Alice.

[1]In Law et al. [1998] and IEEE [2000], the mapping is defined in the elliptic curve case as $\mu(T) = 2^m + (x(T) \bmod 2^m)$, where $x(T)$ is the x-coordinate of the point T, converted to an integer. In the multiplicative group case modulo a prime, also specified in IEEE [2000], it is defined as $\mu(T) = 2^m + (T \bmod 2^m)$. However, the specific mapping μ does not affect the attack in this article.

3. Alice checks that the ephemeral public key R_B is in the group G and that $R_B \neq O$, and computes a group element K_{AB} as
$$K_{AB} = hs_A S_B,$$
where $s_A = (r_A + \mu(R_A)w_A) \bmod n$ and $S_B = R_B + \mu(R_B)W_B$. If $K_{AB} = O$, Alice rejects it; otherwise she accepts it as the shared secret key.
4. Bob checks that the ephemeral public key R_A is in the group G and that $R_A \neq O$, and computes a group element K_{BA} as
$$K_{BA} = hs_B S_A,$$
where $s_B = (r_B + \mu(R_B)w_B) \bmod n$ and $S_A = R_A + \mu(R_A)W_A$. If $K_{BA} = O$, Bob rejects it; otherwise he accepts it as the shared secret key.

If Alice and Bob follow the protocol, they will compute the same shared secret key:
$$K_{AB} = hs_A S_B = hs_A s_B P = hs_B S_A = K_{BA}.$$
The values s_A and s_B are called *implicit signatures* (see Law et al. [1998 Section 5.2]) as they are intended to authenticate the corresponding ephemeral public key R_A or R_B. The scalar multiplication by the cofactor h, also called *cofactor multiplication*, prevents small-subgroup attacks [Menezes et al. 1995a; Jablon 1996; van Oorschot and Wiener 1996; Lim and Lee 1997].[2] (See Kaliski, Jr. [1998] for a version of cofactor multiplication that can interoperate with implementations that do not implement cofactor multiplication.)

The basic protocol is attractive in several regards: it provides implicit key authentication and forward secrecy with respect to both parties; it is computationally efficient; it is bandwidth efficient; and it involves only group operations and a simple mapping. Each party's computational cost is roughly 2.5 scalar multiplications—one for generating the ephemeral key pair, another for the scalar multiplication by s_A or s_B, and the remaining half for the scalar multiplication by $\mu(R_A)$ or $\mu(R_B)$—plus the cost of cofactor multiplication. The cost of cofactor multiplication depends on the group. For elliptic curve cryptosystems, it is not significant since the cofactor is typically small. The bandwidth requirement per party is one ephemeral public key (plus the overhead for the certificates).

By comparison, without optimization, "unified" Diffie–Hellman [Blake-Wilson et al. 1997], MTI/A0 [Matsumoto et al. 1986], and the Goss protocol [Goss 1990] all cost about three scalar multiplications, and the Station-to-Station protocol [Diffie et al. 1992] costs about two scalar multiplications plus one signature generation and one signature verification. Among these, only unified Diffie–Hellman and the STS protocol, like MQV, offer forward secrecy with respect to compromise of both parties' static private keys; MTI/A0 offers forward secrecy with respect to one party or the other but not both. On the other hand, MTI/A0 and the Goss protocol, like MQV, involve only group operations; the others involve a hash function or signature scheme operations.

Two variants of the basic protocol are also given in Law et al. [1998]. Protocol 2 is a one-pass protocol intended for key establishment in store-and-forward

[2] In IEEE [2000], cofactor multiplication is optional.

environments such as electronic mail. It is the same as Protocol 1 except that the recipient's static and ephemeral key pairs are the same and Step 2 is omitted, thereby eliminating one pass and letting the recipient be offline. Protocol 3 is a three-pass protocol that includes key confirmation. It adds the following steps to Protocol 1.

5. Alice derives a key κ' from the shared secret key K_{AB} and computes
$$t_A = \mathrm{MAC}_{\kappa'}(2, B, A, R_B, R_A),$$
where MAC is a message authentication code and B and A are identifiers for Bob and Alice. She sends t_A to Bob.
6. Bob derives a key κ' from the shared secret key K_{BA} and checks that the value t_A is correct. If not, Bob rejects the shared secret key. Bob then computes
$$t_B = \mathrm{MAC}_{\kappa'}(3, A, B, R_A, R_B)$$
and sends t_B to Alice.
7. Alice checks that the value t_B is correct. If not, Alice rejects the shared secret key.

(Adding these steps increases the number of passes to four. To reduce to three, Step 5 can be merged into Step 3 of the basic protocol, and Step 6 into Step 4.)

The basic protocol is in IEEE Std 1363-2000 [IEEE 2000], specified as a key agreement scheme to which key confirmation may be added. Both the basic protocol and the variant with key confirmation are in the ANSI X9.63 draft standard for key establishment based on elliptic curve cryptography [ANSI 2001].

3. AN UNKNOWN KEY-SHARE ATTACK

A simple way Eve might attempt to mount an unknown key-share attack is to obtain a certificate associating her name with Alice's public key. If she replaces Alice's certificate with her own in a key agreement protocol, Bob may believe he is agreeing on the key with Eve, although he actually shares it with Alice.

This simple attack is an example of source substitution, and it motivates the *proof of possession* requirement in typical certificate request protocols (see, e.g., Adams and Farrell [1999, Section 2.3]): the requester must know the private key to obtain a certificate for the corresponding public key. In principle, the certificate authority could check for duplicate public keys, but this is impractical if a large number of certificate authorities are involved.

Eve must thus obtain a certificate for a *new* public key W_E such that she knows the corresponding private key w_E, and such that Bob will compute the same shared secret key given Eve's public key as he would have given Alice's. The following "protocol hijacking" attack on a protocol run from Alice to Bob accomplishes these goals.

1. Eve intercepts Alice's ephemeral public key R_A, en route to Bob.
2. Eve selects an integer u between 1 and $n-1$ and computes an ephemeral public key R_E as
$$R_E = R_A + \mu(R_A)W_A - uP.$$

(Note that Eve does not know the corresponding private key r_E.) If $R_E = O$, Eve repeats this step with another integer u.

3. Eve computes a static key pair (W_E, w_E) as
$$\begin{aligned} w_E &= \mu(R_E)^{-1} u \bmod n \\ W_E &= w_E P. \end{aligned}$$
4. Eve obtains a certificate for her static public key W_E. Since she knows the corresponding private key w_E, she can satisfy the required proof of possession.
5. Eve initiates a protocol run with Bob, identifying herself as Eve, and sends her ephemeral public key R_E.
6. Eve receives Bob's ephemeral public key R_B and forwards it to Alice.

Under this attack, Alice will verify Bob's certificate and compute the shared secret key K_{AB} as usual, while Bob will obtain and verify *Eve's* certificate and compute a shared secret key K_{BE} as

$$K_{BE} = hs_B S_E,$$

where s_B is as previously defined and $S_E = R_E + \mu(R_E)W_E$. Bob's key will be the same as he would have computed with Alice:

$$\begin{aligned} hs_B S_E &= hs_B(R_E + \mu(R_E)W_E) \\ &= hs_B(R_A + \mu(R_A)W_A - uP + \mu(R_E)w_E P) \\ &= hs_B(R_A + \mu(R_A)W_A - uP + \mu(R_E)(\mu(R_E)^{-1} u \bmod n)P) \\ &= hs_B(R_A + \mu(R_A)W_A) \\ &= hs_B S_A \\ &= K_{BA}. \end{aligned}$$

Eve must obtain the certificate for her static public key *during* Alice's protocol run, so Alice may notice a delay between the time she sends her ephemeral public key and the time she receives Bob's. But as certificate authorities are increasingly providing online services, this is not necessarily a problem for Eve. Protocol 2 of Law et al. [1998] (the one-pass protocol) is vulnerable even if the certificate authority is offline, since the recipient is assumed to be offline as well.

Note that Eve cannot compute the shared secret key herself since she does not know the ephemeral private key r_E.

Other unknown key-share attacks may be possible depending on how the protocol is implemented. For instance, Eve may be able to attack the initiator (i.e., Alice) as well as the responder (i.e., Bob) depending on when certificates are exchanged and when the parties identify themselves.

Compared to attacks on other key agreement protocols, this attack is much more effective. The attack on the MTI/A0 protocol, for instance, is readily prevented if the certificate authority requires proof of possession of the private key. The attack here succeeds despite such a requirement. (The attack on the STS-MAC protocol [Blake-Wilson and Menezes 1999], like the one here, also needs more defense than just a proof of possession.)

4. COUNTERMEASURES

The primary countermeasure to an unknown key-share attack, which is also a generally recommended element of key establishment protocols, is to include a key confirmation step as in Protocol 3 of Law et al. [1998]. To pass Bob's check, Eve would need to supply a value $t_E = \mathrm{MAC}_{\kappa'}(2, B, E, R_B, R_E)$. But she does not know the key κ' and other authentication tags are not helpful to her since they are based on different inputs. Thus, while Eve can coerce Alice and Bob into the unknown key share, one of them will reject the key with high probability.

Another countermeasure that might be considered is *ephemeral key commitment*: the parties exchange one-way hashes of their ephemeral public keys before exchanging the ephemeral keys (see IEEE [2000, Clause D.5.1]). The mechanics of the exchange are important, as each party needs to be assured that the other party has received the commitment before it "decommits" and sends the ephemeral public key. This assurance can be obtained by appropriate sequencing; for example, Alice sends her commitment, Bob receives it, and sends his commitment, Alice receives Bob's commitment and sends her key; then Bob receives Alice's key and sends his key. Without appropriate sequencing, for instance, if Alice and Bob exchange commitments in parallel, the protocol will still be vulnerable to attack.

Some other countermeasures are as follows.

—*Delay detection.* An alternative that does not involve cryptography is to implement some kind of "delay detection" where a party terminates a run of the protocol if the other party takes too long to reply. Eve's request for a certificate, if it takes a relatively long time to fulfill compared to the other operations in the protocol, might introduce such a delay. However, this physical measure is not applicable in the store-and-forward case.

—*Certificate aging.* The recipient could require a certain "aging" of the certificate. Very recently issued certificates might be considered evidence of an unknown key-share attack and accordingly could be rejected. This is similar to policies in paper-based commerce for rejecting personal checks with low check numbers. While occasionally inconvenient for a new user, aging will block a "real-time" attack as long as the certificate authority is honest and synchronized. Certificate aging is less helpful in the store-and-forward case.

—*Key derivation based on parties' identifiers.* The key κ to be used for subsequent message processing could be derived not only from the shared secret key K_{AB}, but also from the identifiers for Alice and Bob; that is,

$$\kappa = \mathrm{KDF}(K_{AB}, A, B),$$

where KDF is a key derivation function (see ANSI [2001] and IEEE [2000]). Like key confirmation, key derivation is a generally recommended element of key establishment protocols. This approach does not detect an unknown key-share attack, but rather causes the parties to derive different keys when under attack. Attack detection is thus deferred to the actual use of the key.

—*Identification of parties within messages.* A general principle of protocol design is that messages should carry sufficient context to avoid misinterpretation [Abadi and Needham 1996]. Accordingly, messages protected with a

shared secret key should identify the parties involved. Such identification would prevent the misinterpretation of Alice's deposit and Bob's order in the two classic exploitations. If message overhead is a concern, the parties' identifiers could be included in integrity-protection computations without explicitly including the identifiers in the message.

All of the foregoing approaches involve minimal modifications to the protocol, and in contrast to key confirmation and ephemeral key commitment do not involve additional exchanges of data. However, it is important to note that no formal proof of security has been given for any of the proposed modifications to the protocol. While it appears reasonable to claim that the protocol with appropriate countermeasures resists specific attacks, if a proof of security is desired, more extensive modifications may be needed.

5. DISCUSSION

Cryptography has been improving steadily over the past two decades both as a result of experience and through a better understanding of how to prove that a technique is secure. But key agreement protocols remain one of the most challenging areas of research. Indeed, even the definition of what it means for key establishment and even public-key encryption to be secure is still a subject of research nearly 25 years after the concepts were first introduced [Shoup 1999; Bellare et al. 2000]. Thus, it is not surprising that flaws occasionally are found in key agreement protocols, both new and old.

The attack described above on the MQV key agreement protocol is an example of one of those flaws. While its practical impact is minimal—countermeasures are readily available and unknown key-share attacks are in general limited in effectiveness—the attack is an excellent illustration of several principles to be kept in mind when designing any protocol.

5.1 Minor Improvements Can Introduce Weaknesses

The evolution of the MQV protocol is a classic case of the risks of introducing "minor" improvements. In an earlier version, Protocol 1′ of Vanstone et al. [1998] (henceforth, the "original protocol"), the shared secret key is computed (recast in the notation of this article) as

$$K_{BA} = s_B(R_A + \mu'(R_A)\mu'(W_A)W_A),$$

where $s_B = (r_B + \mu'(R_B)\mu'(W_B)w_B) \bmod n$ and μ' is a full-length mapping. To mount an unknown key-share attack against this protocol, Eve would need to solve the equation

$$R_E + \mu'(R_E)\mu'(W_E)W_E = S_A,$$

where $S_A = R_A + \mu'(R_A)\mu'(W_A)W_A$. Because mappings of both W_E and R_E are involved, this appears to be difficult.

Since the value $\mu'(R_A)\mu'(W_A)$ is a full-length value modulo n, the original protocol has three scalar multiplications. Accordingly, it was sought to "improve" the original protocol by replacing $\mu'(R_A)\mu'(W_A)$ with the half-length

value $\mu(R_A)$ (see Law et al. [1998, Section 5.2]). With the mapping of W_A removed, however, the defense against an unknown key-share attack was lost.[3]

The original protocol could have been modified without losing the defense against the unknown key-share attack, for instance by introducing a multiplier of the form $\mu(R_E, W_E)$, where μ is a half-length mapping that takes two arguments. A mapping based on a hash function would probably be adequate and might even lead to a proof of security (see Hirose and Yoshida [1998] for a proposal along these lines, modulo the comments in Baek and Kim [2000]).

Anything simpler, avoiding a hash function in favor of simpler operations, would likely involve ad hoc design decisions. (Should the arguments be added then truncated? Multiplied?)

Interestingly, it would have been possible to improve the efficiency of the original protocol without modifying the protocol at all. Observe that the formula for K_{BA} in the original protocol may be rewritten as

$$K_{BA} = \alpha R_A + \beta W_A,$$

where α and β are defined as

$$\begin{aligned} \alpha &= (r_B + \mu'(R_B)\mu'(W_B)w_B) \bmod n; \\ \beta &= (r_B + \mu'(R_B)\mu'(W_B)w_B)\mu'(R_A)\mu'(W_A) \bmod n. \end{aligned}$$

The key can thus be computed as a sum of two scalar multiplications, which by the well-known "Shamir trick" (see El Gamal [1985] and Menezes et al. [1997, Note 14.91]), costs only slightly more than one scalar multiplication. (A similar observation applies to the basic protocol.) The original protocol could thus be implemented with slightly more than two scalar multiplications total per party, even better than the 2.5 promised by the improvement!

It could be argued that a modification to the protocol was essential to meet the performance requirements of constrained environments where Shamir's trick cannot be implemented. For instance, there might not be enough memory for the intermediate results involved, or the available hardware acceleration might be directed at scalar multiplies. But the more basic question is this: is an improvement to 2.5 scalar multiplications from 3 (less than 20%) worth the risk associated with a protocol modification?

5.2 All Interactions Must Be Considered

The following argument might have been attempted to show that the MQV protocol resists an unknown key-share attack.

---

[3]The variation between the two forms of multiplier—$\mu'(R_A)\mu'(W_A)$ and $\mu(R_A)$—dates back to the earliest versions of other members of the MQV protocol family. For instance, Protocols 1 and 2 in Menezes et al. [1995c], published in April 1995, have multipliers such as $\mu'(R_A)$ and $\mu'(r_A W_B)$ (both are full-length mappings). In contrast, the multipliers in Protocols 1 and 2 of Vanstone et al. [1998], filed in May 1995, are $\mu'(R_A)\mu'(W_A)$ and $\mu'(r_A W_B)\mu'(W_A)$. The variation appears to have no impact on the security of these particular protocols, which are different from both the basic and original MQV protocols in that they include the transmission and verification of the implicit signatures s_A and s_B. However, as already shown, the variation is critical in the basic protocol.

Recall that the parties, Alice and Bob, compute the shared secret key as

$$K_{AB} = hs_A S_B$$
$$K_{BA} = hs_B S_A,$$

where s_A, S_A, s_B, and S_B are as defined previously. Eve's goal is to replace Alice's static public key W_A with her own static public key W_E and to modify one or both ephemeral public keys so that Alice and Bob still compute the same shared secret key as each other, but Bob believes he shares the key with Eve. In other words, Eve seeks new public keys W_E, R'_A, and R'_B such that

$$hs_A S'_B = hs_B S'_A,$$

where $S'_B = R'_B + \mu(R'_B)W_B$ and $S'_A = R'_A + \mu(R'_A)W_E$. Since she cannot control the values s_A and s_B, she must determine S'_A and S'_B satisfying

$$S'_A = cS_A$$
$$S'_B = cS_B$$

for some constant c. Because of the difficulty of the discrete logarithm problem, she must determine the constant first. Eve's solution must therefore consist of the following steps.

1. Select a constant c.
2. Given Bob's static public key W_B, find an ephemeral public key R'_B such that $R'_B + \mu(R'_B)W_B = cS_B$.
3. Given her own static public key W_E, find an ephemeral public key R'_A such that $R'_A + \mu(R'_A)W_E = cS_A$.

Since the equation in Step 2 is nonlinear in R'_B, it is difficult to solve for R'_B unless $c = 1$; in which case $R'_B = R_B$. But even for $c = 1$, it is difficult to solve for R'_A in Step 3. Hence, Eve cannot replace Alice's static public key with her own to produce an unknown key share.

Clearly, the argument just given falls short in view of the attack presented in Section 3. Step 3 is the problem: it assumes that the ephemeral public key R'_A is determined *after* Eve's static public key W_E. This is a natural assumption; after all, Eve needs time to obtain a certificate, and in previous unknown key-share attacks like the one on the MTI/A0 protocol, Eve obtains her static public key before the protocol run.

There is perhaps a more fundamental reason that the argument given above sounds reasonable at first, and it has to do with how protocols are viewed. MQV and other protocols like it are traditionally thought of as *two-party* key establishment protocols, the two parties of course referring to Alice and Bob. The certificate authority is presented as a trusted *third party* (of which there may be more than one) whose participation is abstracted outside the two-party protocol to the role of providing certificates, typically in advance. Consider how the MQV protocol was introduced in Section 2. Before the protocol steps were stated, the input conditions were given:

> "As input to the protocol, Alice has a static key pair (W_A, w_A) and Bob has a static key pair (W_B, w_B), where each key pair is certified and conveyed in a certificate and is presumed to be known and verified by the other party."

This manner of presentation reinforces the view that the certificate authority is offline, not available for interactions during an attack on a protocol run.

As a suggestion for improving protocol descriptions to encourage analysis of all interactions, it would help if the process of obtaining a certificate were included as an explicit step within the protocol, as pointed out by Shoup [1999]. For instance, the first step might have the form, "Step 1. Alice obtains a certificate for her static public key (this need be done only once)." Another reason to include this step explicitly is that the certificate may in fact be obtained from a third party during the protocol, as a "short-lived" certificate [Ellison et al. 1999; Corella 2000]. In addition, an online third party may provide assurance of certificate validity [Myers et al. 1999]. Thus, whether the certificate authority is normally offline or a trusted third-party is online, the protocol covers the full set of interactions, thereby exposing possible unintended interactions that can be exploited by an attacker. Finally, it is worth observing that in key establishment protocols based on symmetric cryptography, the third party is generally a participant in the protocol [Bellare and Rogaway 1995b]. Increasing the visibility of the third party in protocols based on public-key cryptography would provide a more direct correspondence with those protocols.

5.3 Security Requirements Must Be Assessed

One should finally ask whether the unknown key-share attack described in this paper is really an attack at all. The answer depends on the security requirements. If a system does not require a particular property, then does an attack matter? To explore this further, it is helpful to view key agreement protocols at two layers, separating the cryptographic operations from the order in which they are performed. The operations in the MQV protocol (which one might call the "MQV scheme," following IEEE 1363 terminology) include at least:

—domain parameter generation;

—key pair generation; and

—secret value derivation.

The secret value derivation operation takes one party's static and ephemeral key pairs and another party's static and ephemeral public keys as input and produces a key as output. In Alice's case, the operation is defined as

$$\mathrm{SVD}((W_A, w_A), (R_A, r_A), W_B, R_B) = h(r_A + \mu(R_A)w_A)(R_B + \mu(R_B)W_B).$$

The operation is the same for Bob except that his keys and Alice's are interchanged. (This "role-symmetry" is a nice feature of MQV, although it is not required that a secret value derivation operation have this property.) Note that the first party's static public key is not involved in the computation for MQV.

By separating the secret value derivation operation into a different layer, the protocol can be defined in a way that supports more than just MQV. For instance, the MTI/A0 protocol with key confirmation could be obtained by defining the secret value derivation operation as

$$\mathrm{SVD}((W_A, w_A), (R_A, r_A), W_B, R_B) = w_A R_B + r_A W_B.$$

The unified Diffie–Hellman protocol would have

$$\mathrm{SVD}((W_A, w_A), (R_A, r_A), W_B, R_B) = \mathrm{KDF}(w_A R_B, r_A W_B),$$

where KDF is a key derivation function.

It would be helpful to understand more formally what properties are required of the secret value derivation operation for the protocol to meet the security goals in a provable sense. As already observed, the MQV secret value derivation operation itself has an unknown key-share "weakness." But in practice, as long as the protocol includes key confirmation, the weakness is not a concern. Put another way, the security of the protocol *with* key confirmation against unknown key-share attacks does not depend on whether the secret value derivation operation has an unknown key-share weakness.

As a related example from the RSA public-key cryptosystem, consider that the RSA operation itself—the computation $c = m^e \bmod n$, where m is a message, (n, e) is a public key, and c is the resulting ciphertext—has a number of weaknesses due to its multiplicative structure (see Boneh [1999] for a summary). But in practice, the RSA operation is applied in a protocol for which these weaknesses are not a concern. For instance, if the integer m is the result of applying Optimal Asymmetric Encryption Padding (OAEP) [Bellare and Rogaway 1995a] to the actual message, the resulting scheme achieves provable security under the random oracle model despite the multiplicative properties of RSA.

6. CONCLUSIONS

Security protocol designers face the difficult task of reconciling efficiency and security requirements and sometimes must make design decisions that appear well motivated but have unintended consequences. The lessons learned with respect to the MQV protocol illustrate this point, adding to the principles previously espoused for prudent protocol engineering [Abadi and Needham 1996; Anderson and Needham 1995].

Among the possible attacks on a protocol, an unknown key-share attack is one of the least threatening. The attack presented here thus should not discourage use of the MQV protocol per se, as long as appropriate countermeasures are taken. A key confirmation step, for instance, will defeat the attack, and even if that step is not included, the attack is only a threat if parties act on false assumptions about the origin and destination of messages. At no time is the opponent able to decrypt messages.

What we hope the attack will discourage is the development of additional "efficient" protocols that lack a security proof. Recent research and standards development are converging on cryptographic techniques of a variety of types that are provably secure and nearly as efficient as previous ad hoc techniques (for instance, Bellare and Rogaway's [1995a] Optimal Asymmetric Encryption Padding and Probabilistic Signature Scheme [Bellare and Rogaway 1996], which are in IEEE 1363 and the current draft of IEEE P1363a [IEEE P1363 Working Group 2001], respectively). As such techniques are adopted in standards and deployed in practice, cryptanalysis will hopefully yield fewer surprises.

ACKNOWLEDGMENTS

Don Johnson, Alfred Menezes, Jerry Solinas, and Minghua Qu provided helpful comments on the initial write-up of the unknown key-share attack which I reported to the IEEE P1363 working group in June 1998, and John Linn offered comments on the final manuscript. The discussion on ephemeral key commitment is based on conversations with Don Johnson and David Kravitz during the development of IEEE 1363. I also thank the anonymous reviewers for their helpful comments. As is my custom, I also thank God (Col. 3:17).

REFERENCES

ABADI, M. AND NEEDHAM, R. 1996. Prudent engineering practice for cryptographic protocols. *IEEE Trans. Softw. Eng.* (*TSE*) *22*, 1 (Jan.), 6–15.

ADAMS, C. AND FARRELL, S. 1999. Internet X.509 public key infrastructure certificate management protocols. IETF RFC 2510.

ANDERSON, R. AND NEEDHAM, R. 1995. Robustness principles for public key protocols. In *Advances in Cryptology—CRYPTO '95 Proceedings*, D. Coppersmith, Ed., *Lecture Notes in Computer Science* vol. 963, Springer-Verlag, New York, 236–247.

ANSI. 2000. *ANSI X9.63*, *Public Key Cryptography for the Financial Services Industry*: *Key Agreement and Key Transport Using Elliptic Curve Cryptography*. ANSI. Working draft. June 15, 2001.

BAEK, J. AND KIM, K. 2000. Remarks on the unknown key-share attacks. *IEICE Trans. Fund. E83-A*, 12 (Dec.), 2766–2769.

BELLARE, M. AND ROGAWAY, P. 1995a. Optimal asymmetric encryption—How to encrypt with RSA. In *Advances in Cryptology—EUROCRYPT '94 Proceedings*, A. D. Santis, Ed., vol. 950, *Lecture Notes in Computer Science*, Springer-Verlag, New York, 92–111.

BELLARE, M. AND ROGAWAY, P. 1995b. Provably secure session key distribution—The three party case. In *Proceedings of the 27th Annual Symposium on the Theory of Computing* (1995), ACM, New York, 57–66.

BELLARE, M. AND ROGAWAY, P. 1996. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology—EUROCRYPT '96 Proceedings*, U. M. Maurer, Ed., vol. 1070, *Lecture Notes in Computer Science,* Springer-Verlag, New York, 399–416.

BELLARE, M., BOLDYREVA, A., AND MICALI, S. 2000. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology—EUROCRYPT 2000 Proceedings*, B. Preneel, Ed., vol. 1807, Springer-Verlag, New York, 259–274.

BLAKE-WILSON, S. AND MENEZES, A. 1999. Unknown key-share attacks on the Station-to-Station (STS) protocol. In *Public Key Cryptography* (*PKC '99*) *Proceedings*, H. Imai and Y. Zheng, Eds., vol. 1560, *Lecture Notes in Computer Science*, Springer-Verlag, New York, 154–170.

BLAKE-WILSON, S., JOHNSON, D., AND MENEZES, A. 1997. Key agreement protocols and their security analysis. In *Proceedings of the Sixth IMA International Conference on Cryptography and Coding* (*IMA '97*), M. Darnell, Ed., *Lecture Notes in Computer Science*, vol. 1355, Springer-Verlag, New York, 30–45.

BONEH, D. 1999. Twenty years of attacks on the RSA cryptosystem. *Not. Am. Math. Soc.* (*AMS*) *46*, 2, 203–213.

CORELLA, F. 2000. Structured certificates and their applications to distributed systems security. Presented at RSA Conference 2000 (San Jose, Calif., Jan. 16–20).

DIFFIE, W. AND HELLMAN, M. 1976a. Multiuser cryptographic techniques. In *Proceedings of AFIPS National Computer Conference,* 109–112.

DIFFIE, W. AND HELLMAN, M. 1976b. New directions in cryptography. *IEEE Trans. Info. Theor. 22*, 6 (Nov.), 644–654.

DIFFIE, W., VAN OORSCHOT, P., AND WIENER, M. 1992. Authentication and authenticated key exchanges. *Des., Codes Cryptogr. 2*, 2, 107–125.

EL GAMAL, T. 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Info. Theor. 31*, 469–472.

ELLISON, C., FRANTZ, B., LAMPSON, B., RIVEST, R., THOMAS, B., AND YLONEN, T. 1999. SPKI certificate theory. IETF RFC 2693.

GOSS, K. 1990. Cryptographic method and apparatus for public key exchange with authentication. U.S. Patent No. 4,956,865.

HIROSE, S. AND YOSHIDA, S. 1998. An authenticated Diffie–Hellman key agreement protocol secure against active attacks. In *Public Key Cryptography* (*PKC '98*) *Proceedings*, H. Imai and Y. Zheng, Eds., *Lecture Notes in Computer Science,* vol. 1431, Springer-Verlag, New York, 135–148.

IEEE. 2000. *IEEE Std 1363–2000*: *Standard Specifications for Public Key Cryptography*. IEEE.

IEEE P1363 Working Group. 2001. *IEEE P1363a D10* (*Draft Version 10*) : *Standard Specifications for Public Key Cryptography*: *Additional Techniques*. IEEE P1363 Working Group. Working draft. Available from http://grouper.ieee.org/groups/1363/.

JABLON, D. 1996. Strong password-only authenticated key exchange. *Comput. Commun. Rev. 26*, 5 (Oct.), 5–26.

KALISKI, JR., B. S. 1998. Compatible cofactor multiplication for Diffie–Hellman primitives. *Electron. Lett. 34*, 25 (Dec. 10), 2396–2397.

LAW, L., MENEZES, A., QU, M., SOLINAS, J., AND VANSTONE, S. 1998. An efficient protocol for authenticated key agreement. Tech. Rep. CORR 98-05, Department of C&O, University of Waterloo. Also available from http://grouper.ieee.org/groups/1363/.

LIM, C. AND LEE, P. 1997. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in Cryptology—CRYPTO '97 Proceedings*, B. S. Kaliski, Jr., Ed., *Lecture Notes in Computer Science,* vol. 1294, Springer-Verlag, New York, 249–263.

MATSUMOTO, T., TAKASHIMA, Y., AND IMAI, H. 1986. On seeking smart public-key distribution systems. *Trans. IECE Japan E69*, 99–106.

MENEZES, A., QU, M., AND VANSTONE, S. 1995a. Key agreement and the need for authentication. Presented at *Public Key Solutions '95* (Toronto, Nov.).

MENEZES, A., QU, M., AND VANSTONE, S. 1995b. Some new key agreement protocols providing mutual implicit authentication. In *Proceedings of the Second Workshop on Selected Areas in Cryptography* (*SAC '95, Ottawa, May 18–19*), 22–32.

MENEZES, A., VAN OORSCHOT, P., AND VANSTONE, S. 1997. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Fla.

MENEZES, A. J., QU, M., AND VANSTONE, S. A. 1995c. Some new key agreement protocols providing implicit authentication. Manuscript.

MYERS, M., ANKNEY, R., MALPANI, A., GALPERIN, S., AND ADAMS, C. 1999. X.509 Internet public key infrastructure online certificate status protocol—OCSP. IETF RFC 2560.

SHOUP, V. 1999. On formal models for secure key exchange. Tech. Rep. RZ 3120, April, IBM Research Report. Revised version available from http://www.shoup.net/papers/.

VAN OORSCHOT, P. AND WIENER, M. 1996. On Diffie–Hellman key agreement with short exponents. In *Advances in Cryptology—EUROCRYPT '96 Proceedings*, U. M. Maurer, Ed., *Lecture Notes in Computer Science,* vol. 1070, Springer-Verlag, New York, 332–343.

VANSTONE, S., MENEZES, A. J., AND QU, M. 1998. Key agreement and transport protocol with implicit signatures. U.S. Patent No. 5,761,305.

Received July 2000; revised May 2001; accepted May 2001