Photuris: Design Criteria

William Allen Simpson

DayDreamer
Computer Systems Consulting Services
1384 Fontaine
Madison Heights, Michigan 48071
wsimpson@umich.edu
wsimpson@greendragon.com (preferred)

Abstract. Historic look at design principles and requirements for a practical key management communication protocol. Refinement of terms, threat environment and limitations, and necessary features. Dynamic computational time and network round-trip time are well integrated with protocol specification. First use of anti-clogging tokens to defend against resource attacks.

1 Motivation

Photuris[1] is a session-key management protocol featuring authenticated key agreement with confirmation, defense against resource clogging, forward secrecy of the session-keys, and privacy protection for the parties.

Photuris [KS99] was based on currently available tools, by experienced network protocol designers with an interest in cryptography, rather than by cryptographers with an interest in network protocols. Designing and implementing protocols for an open public network requires consideration of a distributed threat environment that includes bandwidth limitations and propagation latency, ubiquitous casual packet snooping and malicious interference, as well as faulty hardware and software implementation errors.

The definitions of protocol features and threats are more stringent than found in recent survey publications [MvOV97]. Distinctions are made between similar failure modes that have different causes to assist in analysis and specifying corrective measures. Other distinguishing terminology is used, with the usual denotative interpretation, conveying concepts that have more colorful terms (such as "spam" and "smurf") in the network operations community.

[1] "Photuris" is the latin name for the firefly. "Firefly" is in turn the name for the USA National Security Administration's (classified) key exchange protocol for the STU-III secure telephone.

Howard Heys and Carlisle Adams (Eds.): SAC'99, LNCS 1758, pp. 226–241, 2000.

2 Fundamental Principles

2.1 End-to-End

The ultimate objective of Internet Security is to facilitate direct Internet Protocol (IP) end-to-end connectivity between sensitive hosts and users over the Internet. Users will rely on Internet Security to protect the confidentiality of the traffic they send across the Internet and depend on it to block unauthorized external access to their internal hosts and networks.

Users must have confidence in every Internet Security component, including key management. Without this confidence, users may erect barriers ("firewalls") that impede legitimate use of the Internet, or forego the Internet entirely.

2.2 Keys

Internet Security must not place any significance on the easily forged IP Source field. It relies instead on proof of possession of secret knowledge: that is, a cryptographic key.

However, secure manual distribution and maintenance of these keys is often cumbersome and problematic. User distribution often leads to long-lived keys, with concommitant opportunity for compromise of the keys.

2.3 Decentralized

Widespread deployment and use of Internet Security is possible through the use of a key management protocol. For example, Kerberos [KN93] can generate host-pair keys for use in Internet Security, much as it now generates session-keys for use by encrypted telnet and other "kerberized" applications.

The Kerberos model has some widely recognized drawbacks. Foremost is the requirement for a highly available on-line Key Distribution Center (KDC), with a database containing every principal's secret-key. This entails significant security risk.

Public-key cryptography [DH76] enables decentralization. Communicating entities can generate session-keys without real-time communication with any third party.

3 Threat Environment

Photuris establishes short-lived session-keys for Internet nodes that frequently access or are accessed by a large and unpredictable number of other nodes. In addition to stationary wired land-line installations, Photuris is intended to support mobile, wireless and satellite environments. Common activities include creating virtual private networks over the common public Internet, transient connections for mobile users and networks operating over bandwidth-limited links, and commercial transactions between numerous clients and servers.

3.1 Delivery of Messages

The Internet Protocol [Pos81] implements best-effort datagram delivery. Datagrams can be damaged, discarded, or duplicated. Successive datagrams may take different paths through the internetwork, resulting in differences in round trip timing, and re-ordering of the datagrams.

3.2 Eavesdropping

The Internet model of operation [Pos81] assumes that nodes listen to each other on their local network, and that intermediate nodes carry traffic between such networks. Every message will be passively monitored by many other parties.

3.3 Interdiction

An interceptor might selectively prevent the transmission of a correct message from one party to another.

3.4 Modification

An interceptor might selectively prevent the transmission of a correct message from one party to another, and modify the message, before sending it to the latter party. This is sometimes called a "Monkey In The Middle" (MITM).

3.5 Races

An interloper can observe the passage of a message from one party to another, and quickly send a bogus message (to either party), before the next correct message arrives (from the other party). When the transmission path of successive datagrams can vary, this race condition is unpredictable.

3.6 Reflection

An interloper can observe the passage of a message from one party to another, extract important fields, and send another message with those fields to that originating party.

3.7 Replay

An interloper can observe the passage of a message from one party to another, extract important fields, and send another message with those fields to the latter party.

3.8 Resource Clogging

The easiest and most common attack experienced in the Internet is an excessive number of datagrams sent to a target network or node. Processing these datagrams can exhaust the computing resources of a target node. This form of attack often has a randomized, forged IP Source.

3.9 Resource Flooding

A large number of datagrams might exceed the available bandwidth of a link, preventing the passage of legitimate datagrams. This form of attack usually has a sub-network broadcast IP Source. These are difficult to distinguish without knowledge of the specific topology of the (distant) sub-network.

4 Threat Limitations

Internet Security is not a panacea. It is not intended to prevent or recover from all possible security threats. Rather, it is designed to protect against most probable and feasible attacks.

In particular, non-cryptographic attacks are outside the scope of this document. For example, cutting a link, jamming a radio signal, or tampering with a computer device might be important security threats, but are not within the province of a key management protocol.

By the very nature of a key management protocol, the threat of *Interdiction* (3.3) can be reduced to a non-cryptographic attack. Prevention of key management traffic is no more harmful than prevention of normal data traffic. In a secure environment, no normal traffic will flow without successful key agreement.

The *Resource Flooding* (3.9) denial of service attack (exceeding the bandwidth of a link) is another non-cryptographic attack. These infrastructure attacks are best dealt with through other means, such as [BCC+98]. However, the use of Internet Security firewalls around vulnerable links (between links of significantly different bandwidth) can change the effect of the attack to *Resource Clogging* (3.8), where Internet Security provides a tractable solution.

4.1 Anonymity

Internet Security is not expected to function in an environment where the identities of the principals are concealed from each other. Authenticated key agreement is antithetical to promiscuously accepting "anonymous" null and/or unverifiable identities.

The effectiveness of any provision of anonymity is unknown. Some folks have asserted that traffic analysis is sufficiently thorough to determine the parties to any transaction. Unfortunately, thus far these analysts have refused to give concrete details.

4.2 Multicast

Key management is more difficult in a multicast environment. The IP Destination indicates a potentially large and disparate group, rather than a single node.

Senders to a multicast group may share common a Security Parameters Index (SPI), when all communications are using the same security configuration parameters. In this case, the receiver only knows that the message came from a node knowing the SPI for the group, and cannot authenticate which member of the group sent the datagram.

Multicast groups may also use a separate SPI value for each IP Source. As each sender is keyed separately, data origin authentication is also provided.

A participating node is not necessarily in control of the SPI selection process. A single node or cooperating subset of the multicast group may work on behalf of the entire group to set up a Security Association.

It is anticipated that Photuris would be used first to establish a distribution SPI and session-key, and that another orthogonal key distribution mechanism will use that SPI to send the group keys. This is a matter for future research. Such mechanisms are outside the scope of this document.

4.3 Multi-user Hostility

Internet Security protects against threats that come from the external network, not from mutually suspicious users of the nodes themselves. In essence, this is another non-cryptographic mode of attack that warrants further elaboration:

- A secure multi-user operating system is able to protect its resources from hostile users, and prevent one hostile user from damaging the resources controlled by another user.
- A secure multi-user operating system incorporates strong support for user-oriented discretionary access controls.
- If the operating system has any security vulnerability, such that internal information may be revealed or the information of one user may be inadvertently disclosed to another user, then there is no basis for separate user-oriented key management.

It has been suggested that the Photuris exchange could also be established between particular application or transport processes associated with a user of a node. This is a matter for future research. Such mechanisms are outside the scope of this document.

Successful use of application, transport or user-oriented keying requires a significant level of operating system support. Use of multi-user segregated exchanges likely requires added functionality in the transport API of the implementation operating system. Such mechanisms are emphatically outside the scope of this document.

5 Design Requirements

The fundamental role of this key management protocol is to verify the values exchanged, while ensuring that the resulting keys are not known by another party.

5.1 Strength

It is required that it be computationally infeasible for any unintended party to discover the mutually computed shared-secret during the lifetime of the key management exchange.

While it might seem obvious that a computer security protocol must be computationally secure, this requirement defines the extent of its cryptographic strength. That is, the minimal requirement is related to the lifetime of the ephemeral exchange itself, although other goals might extend the desired lifetime. There is no requirement that the strength be effectively infinite. Typical exchange lifetimes are measured in minutes or hours.

When coupled with the features described later, this minimal requirement imposes conservative design limitations on the exchange messages and key derivation techniques. While it is preferable that the strongest available method be used, a light-weight requirement allows the protocol to be securely deployed in cellular telephones, and other low computational power devices.

In practice, the estimated strength of the computed shared-secret is chosen to match the cryptographic strength of the session-keys for the chosen security parameters. These relative strengths are measured in time rather than entropy. The protocol must transparently support increasing amounts of entropy corresponding to adversary improvements in computational power.

5.2 Confirmation

Explicit confirmation is required for completion of each phase of the protocol.

While it might seem obvious (to an experienced network protocol designer) that the protocol run is not complete until the parties agree it has completed, there abound numerous examples of theoretical key agreement protocols without this important property. Typical network protocols execute a three-way handshake for both initiation and termination of a communication session.

This requirement ensures that the protocol is robust against duplication, loss and re-ordering of messages, and effectively prevents many reflection and replay attacks.

5.3 Authentication and Authorization

Each party must successfully verify the exchanged protocol values before using any resulting keys.

It has been shown [DvOW92] that secure key agreement must be coupled to authentication. Each party needs assurance that an exchanged key is not shared with an imposter.

In addition to ensuring protocol correctness, this requirement allows the resulting keys to be associated with access permissions and authorization policies. When using asymmetric (public/private key-pair) identities, it is possible that an active interception and modification attack will use entirely valid certificates. Operators should be suspicious when the peer identities are all certified by a single entity, such as the regional security agency equivalent. This attack can only be prevented through rigorous authorization policy enforcement.

6 Design Features

Photuris establishes short-lived session-keys between two parties, without passing the session-keys across the Internet. These session-keys directly replace the long-lived secret-keys (such as passwords and passphrases) historically configured for security purposes.

The basic Photuris protocol [KS99] utilizes these existing previously configured secret-keys for identification of the parties. This is intended to speed deployment and reduce administrative configuration changes.

Photuris is independent of any particular party identification method or certificate format. Support for symmetric key party identification is required to be implemented, and asymmetric key party identification is optionally supported by extensions [Sim99].

In addition to establishing session-keys, Photuris is easily capable of generating high quality unpredictable secrets. This facility can be useful to augment or expand lower quality user symmetric secret-keys, and to substitute for computationally expensive asymmetric public/private-key operations.

Photuris has been designed:

- for frequent exchange of limited lifetime session-keys between parties.
- for associating security parameters with these session-keys.
- to thwart certain types of denial of service attacks on node resources.
- to maximize computational efficiency.
- to scale to a large number of networks and nodes.
- to support the use of a variety of authentication methods, and facilitate the exchange of many identification types.
- to protect the privacy of the parties and the associated security parameters.
- to provide these services with minimal administrative configuration and user effort.

6.1 Forward Secrecy

Many security breaches in cryptographic systems have been facilitated by designs that generate traffic session-keys (or their equivalents) well before they are

needed, and then keep them around longer than necessary. This creates many opportunities for compromise, especially by insiders. A carefully designed key management system can avoid this problem.

The rule is to avoid using any long-lived keys (such as a public/private key-pair) to encrypt session-keys or actual traffic. Such keys should be used solely for identification (entity authentication) purposes. Theft of the key used to authenticate key management exchanges might allow the thief to impersonate the party in future exchanges, but itself would not decode any past traffic that might have been recorded.

Session-keys for traffic authentication and encryption should be generated immediately before use, and then destroyed immediately after use, so that they cannot be recovered. Key generation values should not be directly derived from the values of any previous session-keys.

Photuris utilizes cryptographic hashing algorithms for its key generation pseudo-random functions. The initializing data values are carefully arranged to avoid related key analysis.

Session-keys are derived from large, unpredictable data values. At least two of these values are secret:

1. the computed shared-secret. This is based on short-term secret values. In theory, it is possible that the shared-secret could be recovered (computationally) from the publically exchanged values.
2. authentication key(s) associated with the parties. These involve medium to long-term secret values. In practice, it is more likely that the authentication key(s) would be recovered (by theft or coercion) from the parties.

This combination of multiple disparate secret values ensures that computational discovery of session-keys through cryptanalysis of the key management system requires the solution of multiple "hard" problems.

6.2 Perfect Forward Secrecy

Photuris goes to considerable lengths to achieve perfect forward secrecy [Dif90]. When the authentication key(s) are periodically destroyed, and the destruction is sooner than the feasible recovery of the shared-secret, the derived session-keys are not recoverable from the exchange.

This goal raises the desirable strength for the computed shared-secret, to the expected lifetime of the authentication key(s).

6.3 Privacy Masking

Concealing the correspondents from other parties is often desirable for confidential traffic, especially where this would reveal the location of a mobile user. Although each IP datagram carries a cleartext IP Destination, the ultimate destination can be hidden by "laundering" it through an encrypted tunnel. The IP Source could be hidden in the same manner. If the tunnel IP Source has been dynamically allocated, it provides no useful information to an eavesdropper.

Hiding Identities. This leaves the identifying information that the parties send for authentication. The identities can be easily protected using a privacy-key based on the established shared-secret. Message padding conceals variations in identity lengths. This prevents an eavesdropper from learning the identities of the parties, either directly from names in certificates or by checking against a known database of public keys.

> *Nota Bene:* The terminology is a play on words. Masking in computational algorithms is often applied to the hiding or extraction of field values. Masking in social venues is a physical device to hide identity and protect privacy.

This privacy masking is distinguished from party *anonymity* (4.1), where one of the parties refuses to identify itself to the other. Mutual verification of authentication and authorization is fundamental to the security of this protocol.

> *Caveats:* The scheme is not foolproof. By posing as the Responder, an adversary could trick the Initiator into revealing its identity.
>
> The attack requires the adversary to (1) gain access to a physical transmission link and race the Responder, or (2) subvert Internet routing for the same purpose. These attacks are considerably more difficult than passive vacuum-cleaner monitoring. Moreover, unless the adversary can steal the authentication key belonging to the Responder, the Initiator will discover the deception when verifying the exchanged values.
>
> It is not possible for an Initiator to similarly trick the Responder. The Responder will verify the Initiator Identification before returning its own identity.

Inhibiting Cryptanalysis. In addition to more obvious benefits, hiding the message fields inhibits cryptanalysis of session-key generation by reducing the number of known fields.

Also, privacy masking conceals the attributes associated with the visible traffic Security Parameters Index (SPI). Message padding conceals variations in attribute lengths. When multiple transform algorithms are implemented, hiding attribute choices may inhibit traffic cryptanalysis.

Preventing Forgery. In real time transaction environments, such as banking, it can be even more critical that protection be provided against forgery. The confidentiality of the transaction might only be needed for a short period of time, yet protection against forgery will be needed for a relatively long period of time. Hiding the message verification fields prevents an adversary from direct verification of forgery attacks on the authentication function.

However, unlike the Station-To-Station authentication protocol [DvOW92], the security of the message exchange is not dependent on hiding of the verification fields. Instead of unilateral signatures over public values, Photuris uses

keying material contributed by both parties. In effect, the derived verification-keys are session-keys for the exchange, and share the property of multiple "hard" problems.

6.4 Resource Defenses

Protecting sensitive data against compromise while in transit over the Internet is necessary, but not sufficient. The network and computing resources themselves must also be protected against unauthorized access, malicious attack or sabotage.

To grant access to authorized users regardless of location, it must be possible to cheaply detect and discard bogus datagrams. Otherwise, an adversary intent on sabotage might rapidly send datagrams to exhaust the node's CPU or memory resources.

Using Internet Security authentication facilities, when a datagram does not pass an authentication check, it can be discarded without further processing. This is easily done with manual (null) session-key management between trusted nodes at relatively little cost, given the speed of cryptographic hashing functions compared to public-key algorithms.

Unfortunately, such a trusted node will have only a fixed number of keys available. These keys will tend to have long lifetimes. This entails significant security risk.

Automatic key management is necessary to generate short-lived session-keys between parties. But, there is a potential Achilles heel in the key management protocol.

Because of the use of CPU-intensive operations such as modular exponentiation, key management schemes based on public-key cryptography are vulnerable to *Resource Clogging* (3.8). Although a complete defense against such attacks is impossible, Photuris features make them much more difficult. Resistance is accomplished with multiple, successive, inter-dependent layers.

Anti-clogging Tokens. Path validation is achieved through the exchange of unique "cookies" in the first phase. These tokens are included as an exchange identifier (M_{IR}) in every subsequent message.

This *Cookie Exchange* (A.1) provides a weak form of message origin authentication and verifies the presence of network communications between the parties, thwarting the saboteur from using random IP Source addresses. The simple validation of these cookies uses the same level of resources as other Internet Security authentication mechanisms.

This forces the adversary to (1) use its own valid IP address, or (2) gain access to a physical transmission link and appropriate a range of IP addresses, or (3) subvert Internet routing for the same purpose. The first option allows the target to detect and filter out such attacks, and significantly increases the likelihood of identifying the adversary. The latter two attacks are considerably more difficult than merely sending large numbers of datagrams with randomly chosen IP Source addresses from an arbitrary point on the Internet.

Caveats: The cookie exchange does not protect against an interloper that can race to substitute another cookie (3.5), or an interceptor that can modify and replace a cookie (3.4). As noted earlier, these attacks are considerably more difficult than passive vacuum-cleaner monitoring. Moreover, unless the adversary can steal the authentication key belonging to the Responder, the Initiator will discover the deception when verifying the exchanged values.

Exchange Identifier. The message exchange identifier (M_{IR}) consists of an ordered pair (both cookies). The Responder cookie (c_R) is dependent on a time-variant secret (Kc_R), the Initiator cookie (c_I), an anti-replay exchange counter (C'), and other implementation specific factors. It should not be possible to successfully *Reflect* (3.6) or *Replay* (3.7) an earlier cookie from either party.

Exchange State. In addition to the obvious benefits, path validation inhibits exhaustion of memory and storage resources. No storage state is created in the Responder until after a successful three-way handshake.

Validating Messages. Initial integrity checking of every message is provided by the UDP [Pos80] length and checksum, inhibiting casual message *Modification* (3.4). In the later phases of the exchange (A.3, A.4, A.5), the combination of the privacy mask with checking of the message padding values prevents appending modification. Chaining of successive verification values in calculation of message validation and resulting session keys aids in preventing *Reflection* (3.6) and *Replay* (3.7).

6.5 Scalability

A common predilection in the theoretical cryptological community is an expressed desire to eliminate "interactiveness", and otherwise minimize the number of messages between parties. That appears to arise from the unwarranted assumption this would reduce the opportunity for interference.

However, in the Internet Security environment, an adversary that can interfere with any message can probably interfere with all of the messages. The key management protocol can only protect against late comers through verification of the whole message exchange.

Pacing Messages. Interactivity can distribute computations over time, utilizing inherent latency associated with geographic network topology. For a local network, there are few nodes with low latency between them. As the network environment expands, so does the round-trip time of the message exchanges, affording an opportunity to employ larger computational effort between passes, or to support a larger number of nodes with the same effort.

In Photuris, each computationally expensive operation involves a separate message. As computational or network resources are available, the message pacing naturally varies, and prevents synchronization between multiple Photuris exchanges.

Reduced Computation. In addition to the obvious benefits, this arrangement grants an opportunity for pre-computation of a public-value to be used in multiple closely spaced Photuris exchanges. The pre-computed value can be sent immediately, allowing parallel computation of the resulting shared-secret during the round-trip time.

6.6 Simplicity

The hallmark of successful Internet protocols is that they are relatively simple. This aids in analysis of the protocol design, improves implementation interoperability, and reduces operational considerations.

In Photuris, each message has a single purpose. Message fields are organized in the order that they are processed. Similar message fields appear in the same order in each message.

No more than one optional feature is included in any message, and such options are listed at the end of the message. The format of options is the same as in other Internet protocols, so that implementation code is familiar and can be re-used from other projects.

Although abundant combinations of algorithms offer great flexibility, only a few have been selected for inclusion in the underlying protocol. Choosing these selected schemes in advance allows intensive review of characteristics and potential interactions. This analysis can promote confidence in the security of the implementations.

7 Conclusion

Photuris provides a scalable solution for session-key management. Comprehensive resource defenses ensure that deployment is robust. Provision of privacy masking and forward secrecy raise a strong barrier against cryptanalysis of the key management system.

The distinguishing terminology developed here is used to clarify *"Photuris: Design Rationale"*. Elaboration on the design of the messages will be found there.

A Message Summary

In Photuris, the traditional Alice (A) and Bob (B) are called the Initiator (I) and Responder (R) instead. The following sections describe an exchange where both parties have asymmetric keys, resulting in a pair of secret identities and associated symmetric secret-keys.

When the parties already have existing secret keys (pre-configured or generated by an earlier exchange), the *Secret Exchange* may be omitted.

The *Secret Exchange* and *SPI Messages* may also flow in the other direction (from Responder to Initiator). Only the Initiator to Responder form is illustrated. See [KS99] and [Sim99] for further details.

A.1 Cookie Exchange

The Initiator begins the exchange. The Responder provides an exchange counter (C'), a list of available exchange schemes (So), and a unique exchange identifier (M_{IR}).

(1) $$I \rightarrow R : c_I, 0, C$$

(2) $$I \leftarrow R : M_{IR}, 1, C', So$$

C = previous counter (or zero)
C' = assigned counter (usually $C' = C + 1$)
c_I = Initiator Cookie
$c_R = H(Kc_R, c_I, C', So, InitiatorIPSource, \ldots)$
Kc_R = Responder cookie secret
$M_{IR} = c_I \| c_R$
So = list of offered schemes

A.2 Value Exchange

The Initiator selects a scheme (Ss), and completes the initial three-way handshake by returning the correct counter (C') and exchange identifier (M_{IR}). The parties also exchange their public values (g^x, g^y) and lists of available attributes (Ao_I, Ao_R). A shared-secret (g^{xy}) is calculated.

(3) $$I \rightarrow R : M_{IR}, 2, TBV_I, g^x, Ao_I$$

(4) $$I \leftarrow R : M_{IR}, 3, TBV_R, g^y, Ao_R$$

Ao_I = Initiator list of offered attributes
Ao_R = Responder list of offered attributes
Ss = scheme selected from So
$TBV_I = C' \| Ss$ Initiator three byte value
TBV_R = zero (reserved) Responder three byte value

$VV_I = TBV_I \| g^x \| Ao_I \| TBV_R \| g^y \| Ao_R \| So$
$VV_R = TBV_R \| g^y \| Ao_R \| TBV_I \| g^x \| Ao_I \| So$

A.3 Secret Exchange (Optional)

The parties exchange public keys (K_I, K_R), and secret nonces ($k_{I \leftarrow R}, k_{R \leftarrow I}$) encrypted in those keys. These nonces are combined with the current shared-secret to make high quality symmetric secret keys (Ku_I, Ku_R) to be used in current and future *Identity Exchanges*.

(5) $$I \rightarrow R : M_{IR}, 6, PSILT_I, PSI_I, E_{Kp''_I}(v''_I, K_I, Mp''_I)$$

(6) $$I \leftarrow R : M_{IR}, 5, PSILT_R, PSI_R, E_{Kp''_R}(v''_R, X_I, K_R, Mp''_R)$$

Mp''_I = Initiator message padding
Mp''_R = Responder message padding
PSI_I = Initiator Party Secret Index
PSI_R = Responder Party Secret Index
$PSILT_I$ = Initiator PSI LifeTime
$PSILT_R$ = Responder PSI LifeTime
$v''_I = MAC_{Kv''_I}(M_{IR}, 6, PSILT_I, PSI_I, K_I, Mp''_I, VV_I)$
$v''_R = MAC_{Kv''_R}(M_{IR}, 5, PSILT_R, PSI_R, X_I, K_R, Mp''_R, VV_R)$

$Kp''_I = H(g^x, g^y, M_{IR}, 6, PSILT_I, PSI_I, [g^{xy}])$ Initiator privacy key
$Kp''_R = H(g^y, g^x, M_{IR}, 5, PSILT_R, PSI_R, [g^{xy}])$ Responder privacy key
$Kv''_I = H(g^{xy})$ Initiator verification key
$Kv''_R = H(v''_I, g^{xy})$ Responder verification key

K_I = Initiator public key
K_R = Responder public key
$U_I = PSI_I \| v''_R$ Initiator party symmetric identity
$U_R = PSI_R \| v''_R \| PSI_I$ Responder party symmetric identity
$X_I = E_{K_I}(k_{I \leftarrow R})$
$X_R = E_{K_R}(k_{R \leftarrow I})$

$Ku_I = H(M_{IR}, 6, PSILT_I, PSI_I, 5, PSILT_R, PSI_R, k_{I \leftarrow R}, k_{R \leftarrow I}, g^{xy})$
$Ku_R = H(M_{IR}, 5, PSILT_R, PSI_R, 6, PSILT_I, PSI_I, k_{R \leftarrow I}, k_{I \leftarrow R}, g^{xy})$

A.4 Identity Exchange

The parties verify their identities by proving knowledge of the symmetric secrets, and select attributes (As_I, As_R) for the generated session-keys (Ks_I, Ks_R).

(7) $I \rightarrow R : M_{IR}, 4, SPILT_I, SPI_I, E_{Kp'_I}(X_R, v'_I, As_I, Mp'_I)$

(8) $I \leftarrow R : M_{IR}, 7, SPILT_R, SPI_R, E_{Kp'_R}(U_R, v'_R, As_R, Mp'_R)$

As_I = Initiator list of attributes selected from Ao_R
As_R = Responder list of attributes selected from Ao_I
Mp'_I = Initiator message padding
Mp'_R = Responder message padding
SPI_I = Initiator Security Parameters Index
SPI_R = Responder Security Parameters Index
$SPILT_I$ = Initiator SPI LifeTime
$SPILT_R$ = Responder SPI LifeTime
$v'_I = MAC_{Kv'_I}(M_{IR}, 4, SPILT_I, SPI_I, U_I, As_I, Mp'_I, VV_I)$
$v'_R = MAC_{Kv'_R}(M_{IR}, 7, SPILT_R, SPI_R, U_R, v'_I, As_R, Mp'_R, VV_R)$

$Kp'_I = H(g^x, g^y, M_{IR}, 4, SPILT_I, SPI_I, [g^{xy}])$ Initiator privacy key
$Kp'_R = H(g^y, g^x, M_{IR}, 7, SPILT_R, SPI_R, [g^{xy}])$ Responder privacy key
$Kv'_I = H(Ku_I, g^{xy})$ Initiator verification key
$Kv'_R = H(Ku_R, g^{xy})$ Responder verification key

$Ks_I = H(M_{IR}, Ku_I, Ku_R, v'_I, [g^{xy}])$ Initiator session key
$Ks_R = H(M_{IR}, Ku_R, Ku_I, v'_R, [g^{xy}])$ Responder session key

A.5 SPI Messages (Optional)

Either party may request another set of attributes at a later time, or provide another session-key (Ks_u) to quickly replace one that is expiring.

(9) $I \rightarrow R : M_{IR}, 8, SPILT_n, SPI_n, E_{Kp_n}(v_n, As_n, Mp_n)$
(10) $I \leftarrow R : M_{IR}, 9, SPILT_u, SPI_u, E_{Kp_u}(v_u, As_u, Mp_u)$

As_n = Needed list of attributes
As_u = Update list of attributes
Mp_n = Needed message padding
Mp_u = Update message padding
SPI_n = zero (reserved)
SPI_u = Update Security Parameters Index
$SPILT_n$ = non-zero random
$SPILT_u$ = Update SPI LifeTime
$v_n = MAC_{Kv'_I}(M_{IR}, 8, SPILT_n, SPI_n, v'_I, v'_R, As_n, Mp_n)$
$v_u = MAC_{Kv'_R}(M_{IR}, 9, SPILT_u, SPI_u, v'_R, v'_I, As_u, Mp_u)$

$Kp_n = H(g^x, g^y, M_{IR}, 8, SPILT_n, SPI_n, [g^{xy}])$ Needed privacy key
$Kp_u = H(g^y, g^x, M_{IR}, 9, SPILT_u, SPI_u, [g^{xy}])$ Update privacy key

$Ks_u = H(M_{IR}, Ku_R, Ku_I, v_u, [g^{xy}])$ Update session key

Acknowledgments

Phil Karn was principally responsible for the design of the protocol phases, particularly the "cookie" anti-clogging defense, based on network security protocol implementation experience spanning more than 4 years. In 1994, he provided much of the basic design philosophy text, and developed the initial testing implementation.

In 1995, some months after the first working drafts were distributed, this protocol was discovered to have several elements in common with the Station-To-Station authentication protocol [DvOW92]. Private messages with the authors refined the design criteria, although significant differences in emphasis remain.

This paper was originally part of a larger work. For reasons of space limitation, the more detailed latter two-thirds will be forthcoming in *"Photuris: Design Rationale"*.

References

BCC+98. B. Braden, D. Clark, J. Crowcroft, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. Internet RFC–2309, April 1998.

DH76. W. Diffie and H.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.

Dif90. Whitfield Diffie. Authenticated Key Exchange and Secure Interactive Communication. In *Proceedings of Securicom '90*, March 1990.

DvOW92. Whitfield Diffie, Paul C. van Oorshot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

KN93. J. Kohl and B. Neuman. The Kerberos Network Authentication Service (V5). Internet RFC–1510, September 1993.

KS99. Philip R. Karn and William Allen Simpson. Photuris: Session-Key Management Protocol. Internet RFC–2522, March 1999.

MvOV97. Alfred J. Menezes, Paul C. van Oorshot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

Pos80. Jon Postel. User Datagram Protocol. Internet RFC–768, August 1980. STD–6.

Pos81. Jon Postel. Internet Protocol. Internet RFC–791, September 1981. STD–5.

Sim99. William Allen Simpson. Photuris: Secret Exchange. Work In Progress, March 1999. draft-simpson-photuris-secret-01.txt.