Compact and Unforgeable Key Establishment over an ATM Network

Yuliang Zheng
Monash University, Australia, Email: yzheng@fcit.monash.edu.au
Hideki Imai
The University of Tokyo, Japan, Email: imai@imailab.iis.u-tokyo.ac.jp

March 1998

Abstract Authenticated session key establishment is a central issue in network security. This paper addresses a question on whether we can design a compact, efficient and authenticated key establishment protocol that has the following two properties: (1) each message exchanged between two participants can be transferred in a short packet such as an ATM cell whose payload has only 384 bits, and (2) messages that carry key materials are unforgeable and non-repudiatable without the involvement of a trusted key distribution center. We discuss why the answer to this question is negative if one follows the currently standard approach to key establishment, namely employing secret/public key encryption and, possibly, digital signature. We then present a number of protocols that represent a positive answer to the question. Our protocols are all based on a recently introduced cryptographic primitive called "signcryption" that fulfills both the functions of digital signature and public key encryption with a cost far smaller than that required by "digital signature followed by encryption".

Key Words: ATM Networks, Cryptography, Key Establishment, Multicast, Network Security, Signcryption

1 Introduction

A key establishment protocol is a sequence of specified steps between two or more participants whereby the participants can agree on a shared secret value. The shared secret value is called a *session key*, due to the fact that it is usually used for a single communication session and hence lives for only a relatively short period of time. A major motivation behind session key establishment is to cryptographically eliminate correlations across different communication connections, which would minimize security exposure when a particular session key is compromised. Cryptographic independence of communication sessions would also significantly reduce the risk of replay attacks by an active attacker who has recorded past communication sessions and tries to compromise a current communication session by inserting into it, or replacing (part of) it with, (part of) past sessions. The attack may have or have not compromised the contents of past communication sessions.

A key establishment protocol falls into one of two types. Protocols in the first type rely on shared static keys and use secret key (or symmetric) cryptosystems to ensure the confidentiality of message contents. Although such protocols are generally very efficient, potential problems with them include those associated with the generation and management of static keys. In contrast, protocols in the second type employ public key (or asymmetric) cryptographic techniques. These protocols do not have the problems with static keys, but are not as efficient as those based on secret key cryptosystems.

We are particularly interested in key establishment methods that (1) are efficient, i.e., of a low computational cost, (2) are compact so that a message can be fitted into a small data packet such as a single ATM cell which is composed of a 5-byte header and a 48-byte payload field, and (3) offer message unforgeability and non-repudiation, without the involvement of a trusted key distribution center. To the best knowledge of these authors, none of the public key based protocols in the literature satisfies all the three conditions. A major contribution of this paper is represented by a set of concrete key establishment protocols that all fulfill the three requirements. We also show how to extend the protocols to multicast conference key establishment in which a participant wishes to agree on a common secret key with a multiple number of recipients. We envisage that all these protocols will find applications not only in high speed network layer security solutions, but also in less demanding application layer solutions. The full version of this paper is located at

http://www-pscit.fcit.monash.edu.au/~yuliang/

2 Various Dimensions

There has been an extremely large body of research in the area of key establishment since the publication of the landmark paper by Diffie and Hellman [1], which has resulted in a situation where one may find numerous protocols in the literature, each of which may have different properties. A primary reason behind the emergence of such a large number of key establishment protocols can perhaps be attributed to the many different dimensions of key establishment.

Security — A session key established by an execution of a protocol should be known only to the two participants involved, and also to a KDC or key distribution center if the

protocol involves the KDC. Security of the session key should not be compromised under all the possible attacks that might be encountered in a particular environment where the protocol will be employed. Typical attacks include (1) inferring a session key via (passive) eavesdropping, (2) replaying past messages, (3) interleaving messages from one protocol execution with another, (4) deducing a session key with a known past session key.

Authentication — Entity authentication is a process by which a participant is convinced of the identity of another participant. Entity authentication can be unilateral (one-way) or mutual (two-way). In a mutual authentication protocol, both participants wish to be convinced that the other participant is indeed who he/she claims to be.

A concept that is closely related to and often confused with entity authentication is *identification*. While the aim of identification is similar to entity authentication, namely for one participant, say Alice, to convince another participant, say Bob, of her identity, identification satisfies a more stringent requirement: no participant other than Alice can prove that he or she is Alice, even to him or herself. The difference between entity authentication and identification is made clear by examining a protocol based on a shared static key between Alice and Bob. Alice and Bob can mutually authenticate each other using the static key in three moves or flows [2]. However, such a protocol is not an identification protocol, since whatever produced by Alice using the shared key can also be created by Bob, and vice versa.

Unforgeability and Non-repudiation — In some applications, a participant may require that his or her messages cannot be forged by other participants. Symmetrically, the recipient of a message, especially of one that contains key materials, may require that the sender of the message cannot repudiate at a later stage the fact that he or she is the originator of the message. We envisage that in electronic commerce, non-repudiation and unforgeability of key materials and actual communication sessions that employ a key derived from the key materials may be of particular importance.

Transport v.s. Exchange — We distinguish between two types of key establishment protocols: *key (material) exchange* protocols and *key (material) transport* protocols. Note that key exchange protocols are also called key *agreement* protocols by some researchers. With a key exchange protocol, a shared session key is derived from joint key materials from both participants. Such a protocol requires both participants involved to exchange key materials. In contrast, with a key transport protocol, key materials from which a session key is derived are created by one participant and transferred to the other. A key exchange protocol may be preferred to a key transport protocol in certain applications where a session key is required to be "fair", in that it is dependent on both participants' key materials. However, one should distinguish between key material exchange and shared generation of random numbers as achieved in threshold cryptography [3]. In particular, with a key exchange protocol a participant who is in a position to see, prior to producing his key materials, those from the other participant may control the resultant session key by carefully choosing his key materials. In this sense, a key (material) exchange protocol is essentially the same as a key (material) transport protocol. In general, truly "fair" session key generation cannot be achieved without the involvement of computationally expensive bit/sequence commitment, and hence in these authors' view it should not be set as a goal of key establishment.

Secret v.s. Public Key Cryptosystems — Prior to the execution of a key establishment protocol, two participants may or may not have shared static keys in their hands. In the case of having a shared static key, the most efficient way for them to establish a fresh session key is to use a key establishment protocol built on a secret key (or symmetric) cryptosystem.

On the other hand, if the two participants do not have a shared static key, they may have to use a public key cryptosystem which is not as efficient as a secret key cryptosystem, unless they can ask for help from a key distribution center with whom both participants have a separately shared static key.

Efficiency — Each application may have its own set of requirements on the efficiency of a key establishment protocol. For example, secure mobile communications generally require a "light-weight" protocol, as a mobile device is usually computationally less powerful than a wired one. As a second example, a network layer security application has far more stringent requirements on the efficiency of key establishment than does an upper layer application.

Factors that contribute to the efficiency of a key establishment protocol include (1) the number of moves (or flows, passes) of messages between two participants, (2) the length of messages communicated between the participants (measured in bits), (3) the computational cost invested by both participants, (4) the size of secure storage, (5) the degree of pre-computation (which is especially important if the protocol is intended to be used with computationally weak devices), and so on. One of the challenges that face a protocol designer is to arrive at a key establishment protocol that would not only minimize the first four factors but also maximize the fifth factor, while maintaining the goals the protocol should achieve.

3 Goals and Motivation

The main goals of this research are to design authenticated key establishment protocols that (1) do not rely on a trust key distribution center or KDC, (2) have a low computational cost, (3) are compact so that the length of each message exchanged is as short as possible, and (4) offer unforgeability and non-repudiation.

A practical application that has motivated this research is key establishment at the network layer over an ATM network. As mentioned earlier, only 48 out of the 53 bytes in an ATM cell can be used for transmitting data, as the remaining 5 bytes are reserved for carrying control information. Trans-

mitting a data item of more 384 bits over an ATM network would require two or more ATM cells. While ATM networks are significantly faster than most networks widely used today, transmitting a data item across two or more cells would result in a delay that may not be tolerable in certain high speed applications, primarily due to the necessity of data packetization, buffering, and re-assembling. Therefore, ideally one would like to transmit encrypted key materials in a single ATM cell without the need of splitting data.

In many key transport protocols that rely on secret key cryptosystems, such as those proposed in [4, 5], messages communicated between Alice and Bob are all compact and can be easily fitted into single ATM cells. Some of these protocols do not offer unforgeability or non-repudiation, while the others do so only with the help of a KDC. In other words, these protocols are not suitable for an application where unforgeability and non-repudiation are to be satisfied without relying on a KDC.

Key establishment using public key cryptosystems does not rely on a KDC in achieving unforgeability and non-repudiation. With all currently known public key based key establishment protocols, however, a single payload field of 48 bytes, or of 384 bits, cannot be used to carry unforgeable key materials. To see why this is the case, we take the RSA cryptosystem as an example. In order to maintain a minimal level of security, it is widely believed that the size of an RSA composite should be of at least 512 bits. Thus merely encrypting key materials will result in an expanded outcome that has as many bits as in the RSA composite. (See [6] for a discussion on various data formats for key transport using RSA.) If, in addition, digital signature is involved to achieve unforgeability, the outcome will be even longer. A similar problem occurs with public key cryptographic techniques based on the ElGamal encryption scheme that relies on the discrete logarithm over finite fields.

The ElGamal encryption scheme built on an elliptic curve over a finite field, say $GF(2^{160})$, deserves special attention. With this scheme, a point on the elliptic curve can be compressed so that it occupies only $160 + 1 = 161$ bits. Thus a single ATM cell may be used to transmit un-authenticated key materials of up to about $384 - 161 = 223$ bits. However, a field of 223 bits is too small to carry a key and a time-varying quantity together with a signature. In other words, elliptic curve based public key cryptography does not provide a solution to the problem of compact and unforgeable key establishment.

In the following sections, we show how a recently proposed cryptographic primitive called signcryption can be used to achieve the seemingly impossible goal, namely, to transmit secure and unforgeable key materials in a single ATM cell.

4 Signcryption

A *signcryption* scheme is a cryptographic method that fulfills both the functions of secure encryption and digital signature, but *with a cost smaller than that required by signature-then-encryption*.

An example implementation of signcryption based on the infeasibility of computing discrete logarithm over a large finite field is described below. The example signcryption scheme is called SCS1 and it uses a shortened version of the Digital Signature Standard [7]. The reader is directed to [8, 9] for other example implementations of signcryption.

Let p be a large prime, q a large prime factor of $p-1$, and g an integer with order q modulo p chosen randomly from $[1, \ldots, p-1]$. In addition, we will use E and D to denote the encryption and decryption algorithms of a private key cipher, $hash$ a one-way hash function, and $KH_k(m)$ a keyed hash function/algorithm KH under a key k.

Assume that Alice also has chosen a private key x_a from $[1, \ldots, q-1]$, and made public her matching public key $y_a = g^{x_a} \bmod p$. Similarly, Bob's private key is x_b and his matching public key is $y_b = g^{x_b} \bmod p$.

The example implementation is described in Table 1. Advatages of the signcryption scheme over signature-then-encryption based on RSA are outlined in Table 2.

5 Basic Ideas

Having introduced an example implementation of signcryption in the previous section, now we show how such an implementation allows transportation of key materials in an efficient and compact way. Messages exchanged are so compact that they can all be carried by *a single block whose size is smaller than* $|p|$. We present two possible data formats for Alice to transport key materials to Bob, one carrying directly while the other indirectly key materials.

Direct Transport of Key Materials — The following data format follows from a suggestion made in [8, 9]. We consider a possible combination of parameters: $|p| \geq 512$, $|q| = 160$, and $|KH.(\cdot)| = 80$. For such a choice of parameters, we can transport highly secure and unforgeable key materials of up to 144 bits, in a single ATM cell (48 byte payload + 5 byte header). The actual data from Alice to Bob consist of c, r and s, where $c = E_{k_1}(key, TQ)$, $r = KH_{k_2}(key, TQ, other)$ and $s = x/(r + x_a) \bmod q$, where the key part contained in (key, TQ) may be used directly as a random session key, TQ may contain a time-varying quantity such as a nonce or a time-stamp or both, and *other* may be composed of the participants' identifiers, public key certificates and other supplementary information. It is preferable for E to act as a length-preserving encryption function so that (key, TQ) and $c = E_{k_1}(key, TQ)$ are of the same length.

Note that if key has 64 bits in length, and that TQ requires 32 bits, then $c = E_{k_1}(key, TQ)$ is of 96 bits, and (c, r, s) can be fitted even in a payload that has only $96 + 80 + 160 = 336$ effective bits for data transport. Furthermore, if the quantity TQ is already known to Bob the recipient, then it may be dropped from $c = E_{k_1}(key, TQ)$ to save more positions for transferring key materials.

Indirect Transport of Key Materials — In certain

Signcryption of m by Alice the Sender		**Unsigncryption of (c, r, s) by Bob the Recipient**
$x \in_R [1, \ldots, q-1]$ $(k_1, k_2) = hash(y_b^x \bmod p)$ $c = E_{k_1}(m)$ $r = KH_{k_2}(m)$ $s = x/(r + x_a) \bmod q$	$\Rightarrow \quad c, r, s \quad \Rightarrow$	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ $m = D_{k_1}(c)$ Accept m only if $KH_{k_2}(m) = r$

Table 1: An Example Implementation of Signcryption (SCS1)

security parameters			advantage in	advantage in
$\lvert p\rvert (= \lvert n_a\rvert = \lvert n_b\rvert)$	$\lvert q\rvert$	$\lvert KH_\cdot(\cdot)\rvert$	average comp. cost	comm. overhead
512	144	72	0%	78.9%
1024	160	80	32.3%	88.3%
2048	192	96	59.4%	93.0%
4096	256	128	72.9%	95.0%
8192	320	160	83.1%	97.0%

Table 2: Advantage of Signcryption over RSA based Signature-Then-Encryption

applications, part of a ATM cell payload may be used for other purposes, which leaves no room to accommodate both a random session key and a time-varying quantity. With such a payload structure, we can transport (part of) key materials indirectly. In particular, we may define (c, r, s) as $c = E_{k_1}(TQ)$, $r = KH_{k_2}(TQ, other)$, and $s = x/(r+x_a) \bmod q$. The actual session key may be derived from (k_1, k_2) and other materials, through, for instance, the application of a keyed hash function.

Now assume that TQ has 32 bits. Then we can accommodate (c, r, s) using only $32 + 80 + 160 = 272$ bits. In the case where TQ is already known to Bob, the creation and transmission of the c part can be skipped.

Finally we note that a long TQ, say of 56 bits, may need not be encrypted. However, encryption is mandatory for a short TQ, say of ≤ 40 bits, in order to reduce the risk of replay attacks.

6 Proposals

Now we are ready to describe in full details how to establish fresh random session keys between two participants Alice and Bob, in such a way that all messages exchanged between the two participants are short and computational costs involved are minimized.

6.1 Assumptions

In the following discussions, we assume that system parameters that are common to all participants, and the public and private keys of both Alice and Bob have all been properly set up. In addition, there is a trusted certification authority (CA) that has already issued a public key certificate to each participant. A participant's public key certificate may comply with X.500 certificate format that contains such information as certificate serial number, validity period, the ID of the participant, the public key of the participant, the ID of the CA, the public key of the CA, etc. It would be pointed out that the digital signature scheme used by the CA in creating public key certificates does not have to be one based on ElGamal signature scheme.

Furthermore, we assume that prior to an execution of a key establishment protocol, both participants have already obtained the other participant's public key and its associated certificate issued by the CA, and have checked and are satisfied with the validity of the certificates. The participants may have done so either because they both keep a list of frequently used certificates, or they have obtained and verified the certificates for previous communication sessions.

In describing a key establishment protocol, $key \in_R \{0,1\}^{\ell_k}$ indicates that key is an ℓ_k-bit number chosen uniformly at random. Similarly $NC_b \in_R \{0,1\}^{\ell_n}$ is a nonce chosen by Bob. And TS is a current time-stamp. Typically $\ell_k \geq 64$, $\ell_n \geq 40$, and the number of bits in TS may be decided by the accuracy of clock synchronization, as well as by the life span of a message containing the time-stamp. Finally a 64-bit authentication tag would be long enough for the purpose of key confirmation in most practical applications.

We consider key establishment both through key material transport and exchange.

6.2 Key Transport Protocols

A key transport protocol may use either a nonce or a time-stamp in guaranteeing freshness. The protocol may also transport key materials either directly or indirectly. So there are in total four possible combinations. Table 3 describes two direct key transport protocols, while Table 3 the corresponding two indirect key transport protocols.

The *etc* part may contain data known to both Alice and Bob. Such data may include the participants' names, public keys, public key certificates, protocol serial number, and so on. It may also contain system control information. Note that one of the purposes of sending *tag* is for key confirmation, namely for a participant (Bob) to show the other (Alice) that he does know the new session key. For a less critical application, the time-stamp TS may be transmitted to Bob in clear to further improve the computational efficiency of the protocols.

As can be seen in the tables, protocols that rely on a nonce require one more message than protocols that rely on a time-stamp.

6.3 Key Exchange

In the key transport protocols described above, messages from Bob are not involved the creation of a session key. If one wishes that the session key is generated jointly by Alice and Bob, there are a few different ways that can be used to accomplish this. Here are some examples: (1) $key^* = KH_{key}(NC_b)$, (2) $key^* = KH_{key}(ID_b)$, and (3) $key^* = KH_{key}(NC_b, ID_b)$, where NC_b is a nonce generated by Bob, ID_b is Bob's identifier, and key^* denotes a session key that is jointly determined by information from both Alice and Bob.

Two common properties shared by the four protocols are: (1) Alice identifies herself to Bob (her message to Bob is fresh and unforgeable even by Bob), (2) Bob authenticates himself to Alice if the last response message *tag* is sent (*tag* is fresh and unforgeable by any third party). The protocols can be modified to achieve *mutual identification*: Alice sends to Bob fresh and unforgeable key materials and vice versa.

For two-way communications, Alice and Bob may need to agree upon a pair of random session keys key_1 and key_2. A simple technique is to employ a pseudo-random number generator or a good hashing function to "extend" *key* into (key_1, key_2).

7 Analysis and Comparison

As our key establishment protocols described in Tables 3 and 4 are essentially message transport schemes using signcryption, security of key materials are guaranteed by the security of the signcryption scheme against chosen message attacks [8, 9]. After the successful establishment of a session key, Alice convinces Bob of her identify (the message from Alice is fresh and unforgeable even by Bob). In contrast, Bob can authenticate himself to Alice by sending a response message *tag* which is fresh and unforgeable by a third party (but can be generated by Alice). The four protocols can be modified to achieve mutual identification, at the expense of more computation and message exchanges. Details will be provided in the full version of the paper.

Freshness of a session key is assured through the use of a nonce or a time-stamp. When *tag* is sent, both Alice and Bob are assured that the other participant does know the fresh random session key. The protocols do not rely on a KDC. In addition, key materials transported from Alice to Bob are unforgeable, even by Bob the recipient. The materials are also non-repudiatable by Alice. In an event when Alice denies the fact that she was the person who created certain key materials, Bob can ask for help from a third party called a judge. Bob and the judge may follow a zero-knowledge protocol in settling the dispute [8, 9]. Similar discussions on non-repudiation are applicable to Bob for a modified protocol with mutual identification.

Every message in the key transport protocols proposed in this paper is compact and can be carried by a single ATM cell. In terms of computational cost, it takes one modular exponentiation on Alice's side, and two modular exponentiations on Bob's side which can be reduced to 1.17 exponentiations (on average) when Shamir's method for fast evaluation of the product of several exponentials with the same modulo (see [10]). As for pre-computation, the exponentiation by Alice, $y_b^x \bmod p$, can be done prior to the start of an execution of a protocol, only if Alice knows beforehand that she is going to communicate with Bob at a later time.

Among the key transport protocols based on public key cryptosystems, the one that is most relevant to our protocols is an efficient proposal by Beller and Yacobi [11]. It is assumed that public key certificates have already been transferred prior to an execution of the protocol. In Beller-Yacobi protocol, Alice the sender is assumed to be computationally less powerful than Bob the receiver. Alice uses ElGamal signature scheme to sign a message, and cubic RSA to encrypt the message before delivering it to Bob. Bob holds the matching cubic RSA decryption key and hence can extract the message. The number of modular exponentiations done by Alice is one (for signature generation), and by Bob is four (one for decrypting cubic RSA and three for verifying Alice's digital signature). Shamir's technique for fast evaluation of the product of several exponentials with the same modulo can also be used to reduce two of the exponentiations on Bob's side to 1.17. It is important to note that since the decryption operation for the cubic RSA on Bob's side involves an exponentiation with a full size exponent, it can be very time-consuming, especially when the RSA composite is large. An advantage of Beller-Yacobi protocol over the key transport protocols proposed in this paper is that the modular exponentiation on Alice's side can be fully pre-computed. Table 5 summarizes the comparison between our protocols and Beller-Yacobi protocol.

Next we consider a proposed standard related to security in ATM. The current version of Phase I ATM Security Specification [12] contains two key material exchange protocols. One involves three and the other two moves or flows of messages (see Sections 6.1.1 and 6.1.2 of [12]). These two protocols have been largely based on X.509 [13]. Examining the Specification, we can see that both protocols follow the traditional signature-then-encryption approach, when they are implemented in public key cryptography. As is expected, our protocols based on signcryption are significantly more efficient than the two proposals in the Specification, both in

terms of computational cost and message overhead. A detailed comparison will be included in the full version of this paper.

8 Multicast Conference Key Establishment

The two protocols for direct transport of key materials described in Section 6.2 can be extended to conference key establishment where Alice wishes to establish a common session key with t recipients R_1, R_2, ..., R_t. Such a protocol is very useful in multicast communications. A major difference between a single recipient protocol and a multiple recipient one, both based on signcryption, lies in the length of messages. As shown in previous sections, messages in a key establishment protocol for a single recipient are all compact and can be accommodated in small data packets such as ATM cells. With a protocol for multiple recipients, some messages may be too long to fit in a single ATM cell. Therefore one of our design goals will be use as a small number of cells as possible in transporting key materials.

We assume that each recipient R_i has a unique identifier ID_i, and that the private key of R_i is $x_i \in_R [1, \ldots, q-1]$, and his matching public key is $y_i = g^{x_i} \bmod p$. A multicast conference key transport protocol using nonces is shown in Table 6. The nonces can be replaced with time-stamps, which results in a two-move protocol. A detailed comparison, together with strategies for further improving the efficiency of a multicast conference key transport protocol through randomization, will be included in the full version of this paper.

9 Conclusion

We have presented a number of compact and authenticated key establishment protocols. These are the first protocols based on public key cryptography whose messages can all be carried in very small data packets such as single ATM cells. Our protocols have all been built on the signcryption primitive which fulfills both the functions of signature and encryption in an efficient way. A detailed analysis and comparison has shown that the overall computational cost of these protocols is significantly smaller than all other currently known protocols that are based on public key cryptography.

Acknowledgment

Part of this work was completed while the first author was on sabbatical leave at the University of Tokyo.

References

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 472–492, 1976.

[2] P. Janson, G. Tsudik, and M. Yung, "Scalability and flexibility in authentication services: The KryptoKnight approach," in *Proceedings of INFOCOM'97*. 1997, IEEE.

[3] Y. Desmedt, "Threshold cryptography," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 449–457, 1994.

[4] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryptology - CRYPTO'93*, Berlin, New York, Tokyo, 1993, vol. 773 of *Lecture Notes in Computer Science*, pp. 232–249, Springer-Verlag.

[5] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung, "The KryptoKnight family of authentication and key distribution protocols," *IEEE/ACM Transactions on Networking*, 1995.

[6] D. Johnson and S. Matyas, "Asymmetric encryption: Evolution and enhancements," *CryptoBytes*, vol. 2, no. 1, pp. 1–6, 1996, (available at `http://www.rsa.com/`).

[7] National Institute of Standards and Technology, "Digital signature standard (DSS)," Federal Information Processing Standards Publication FIPS PUB 186, U.S. Department of Commerce, May 1994.

[8] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption)," in *Advances in Cryptology - CRYPTO'97*, Berlin, New York, Tokyo, 1997, vol. 1294 of *Lecture Notes in Computer Science*, pp. 165–179, Springer-Verlag,

[9] Y. Zheng, "Signcryption and its applications in efficient public key solutions," in *Proceedings of 1997 Information Security Workshop (ISW'97)*, Berlin, New York, Tokyo, 1997, Lecture Notes in Computer Science, Springer-Verlag.

[10] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, no. 4, pp. 469–472, 1985.

[11] M. Beller and Y. Yacobi, "Fully-fledged two-way public key authentication and key agreement for low cost terminals," *Electronic Letters*, vol. 30, pp. 999–1001, 1993.

[12] The ATM Forum, "Phase I ATM security specification (draft)," July 1997, ATM Forum BTD-SECURITY-01.03.

[13] ITU, "Information technology - open systems interconnection - the directory: Authentication framework," Recommendation X.509, International Telecommunications Union, 1993.

Direct Key Transport Using a Nonce (Protocol DKTUN)		
Alice		**Bob**
	$\Leftarrow \quad NC_b \quad \Leftarrow$	$NC_b \in_R \{0,1\}^{\ell_n}$
$key \in_R \{0,1\}^{\ell_k}$ $x \in_R [1, \ldots, q-1]$ $(k_1, k_2) = hash(y_b^x \bmod p)$ $c = E_{k_1}(key)$ $r = KH_{k_2}(key, NC_b, etc)$ $s = x/(r + x_a) \bmod q$	$\Rightarrow \quad c, r, s \quad \Rightarrow$	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ $key = D_{k_1}(c)$ Accept key only if $KH_{k_2}(key, NC_b, etc) = r$
verify tag	$\Leftarrow \quad tag \quad \Leftarrow$ (optional)	$tag = MAC_{key}(NC_b)$
Direct Key Transport Using a Time-Stamp (Protocol DKTUTS)		
Alice		**Bob**
$key \in_R \{0,1\}^{\ell_k}$ $x \in_R [1, \ldots, q-1]$ $(k_1, k_2) = hash(y_b^x \bmod p)$ Get a current time-stamp TS $c = E_{k_1}(key, TS)$ $r = KH_{k_2}(key, TS, etc)$ $s = x/(r + x_a) \bmod q$	$\Rightarrow \quad c, r, s \quad \Rightarrow$	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ $(key, TS) = D_{k_1}(c)$ Accept key only if TS is fresh and $KH_{k_2}(key, TS) = r$
verify tag	$\Leftarrow \quad tag \quad \Leftarrow$ (optional)	$tag = MAC_{key}(TS)$

Table 3: Direct Key Material Transport with Signcryption

Indirect Key Transport Using a Nonce (Protocol IKTUN)		
Alice		**Bob**
	$\Leftarrow \quad NC_b \quad \Leftarrow$	$NC_b \in_R \{0,1\}^{\ell_n}$
$x \in_R [1, \ldots, q-1]$ $(k_1, k_2) = hash(y_b^x \bmod p)$ $key = k_1$ $r = KH_{k_2}(key, NC_b, etc)$ $s = x/(r + x_a) \bmod q$	$\Rightarrow \quad r, s \quad \Rightarrow$	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ $key = k_1$ Accept key only if $KH_{k_2}(key, NC_b, etc) = r$
verify tag	$\Leftarrow \quad tag \quad \Leftarrow$ (optional)	$tag = MAC_{key}(TS)$
Indirect Key Transport Using a Time-Stamp (Protocol IKTUTS)		
Alice		**Bob**
$x \in_R [1, \ldots, q-1]$ $(k_1, k_2) = hash(y_b^x \bmod p)$ Get a current time-stamp TS $c = E_{k_1}(TS)$ $r = KH_{k_2}(TS, etc)$ $s = x/(r + x_a) \bmod q$	$\Rightarrow \quad c, r, s \quad \Rightarrow$	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ $TS = D_{k_1}(c)$ Accept (k_1, k_2) only if TS is fresh and $KH_{k_2}(TS, etc) = r$
$key = KH_{k_1,k_2}(TS)$ verify tag	$\Leftarrow \quad tag \quad \Leftarrow$ (optional)	$key = KH_{k_1,k_2}(TS)$ $tag = MAC_{key}(TS, 1)$

Table 4: Indirect Key Material Transport with Signcryption

Protocols	Comp. Cost # of exp.	Pre-Comp. by Alice	Longest Message (typical example)
Beller-Yacobi	$1 + 2.25^*$	Yes	$\geq \lvert n_B \rvert$ bits (≥ 512 bits)
DKTUN & DKTUTS	$1 + 1.17$	Yes$^+$	≤ 384 bits (< 384 bits)
IKTUN	$1 + 1.17$	Yes$^+$	< 384 bits (240 bits)
IKTUTS	$1 + 1.17$	Yes$^+$	< 384 bits (280 bits)

* Including an RSA decryption with a full size exponent.
$^+$ Only when Alice knows whom to communicate with.

Table 5: Comparison with Beller-Yacobi Protocol

Conference Key Establishment Using a Nonce		
Alice and each R_i, $i = 1, \ldots, t$: $NC = NC_1 \oplus \cdots \oplus NC_t$	$\nwarrow$ $\leftarrow$ $\swarrow$ $\begin{pmatrix} ID_1, NC_1 \\ \vdots \\ ID_t, NC_t \end{pmatrix}$ $\nearrow$ $\rightarrow$ $\searrow$	**Each R_i, $i = 1, \ldots, t$:** $NC_i \in_R \{0,1\}^{\ell_n}$
Alice: $key \in_R \{0,1\}^{\ell_1}$ $k \in_R \{0,1\}^{\ell_2}$ $h = KH_k(key, NC, etc)$ $c = E_k(key, h)$ for each $i = 1, \ldots, t$: $v_i \in_R [1, \ldots, q-1]$ $(k_{i,1}, k_{i,2}) = hash(y_i^{v_i} \bmod p)$ $c_i = E_{k_{i,1}}(k)$ $r_i = KH_{k_{i,2}}(h, etc_i)$ $s_i = v_i/(r_i + x_a) \bmod q$	$\rightarrow$ $\begin{pmatrix} c \\ c_1, r_1, s_1 \\ \vdots \\ c_t, r_t, s_t \end{pmatrix}$	**Each R_i:** Find out (c, c_i, r_i, s_i) $(k_{i,1}, k_{i,2})$ $= hash((y_a \cdot g^{r_i})^{s_i \cdot x_i} \bmod p)$ $k = D_{k_{i,1}}(c_i)$ $(key, h) = D_k(c)$ Accept key only if $KH_k(key, NC, etc) = h$ and $KH_{k_{i,2}}(h, etc_i) = r_i$
Alice verifies all tag_1, …, tag_t; **Each R_i** verifies tag_j for each j with $j \neq i$ and $1 \leq j \leq t$.	$\nwarrow$ $\leftarrow$ $\swarrow$ $\begin{pmatrix} ID_1, tag_1 \\ \vdots \\ ID_t, tag_t \end{pmatrix}$ $\nearrow$ $\rightarrow$ $\searrow$ (optional)	**Each R_i:** $tag_i = MAC_{key}(NC, NC_i)$

Table 6: Key Establishment for Multicasting (Basic Protocol)