Published in *ACM Operating Systems Review* 32(4):33–39, 1998.

ID-based Secret-Key Cryptography*

Marc Joye and Sung-Ming Yen

Laboratory of Cryptography and Information Security
Dept of Electrical Engineering, Tamkang University
Tamsui, Taipei Hsien, Taiwan 25137, R.O.C.
E-mail: {joye,yensm}@ee.tku.edu.tw

Abstract. This paper introduces *ID-based secret-key cryptography*, in which secret keys are privately and uniquely binded to an identity. This enables to extend public-key cryptography features at the high throughput rate of secret-key cryptography. As applications, efficient login protocols, an enhanced version of Kerberos, and an ID-based MAC algorithm are presented.
ID-based systems were initially developed in the context of public-key cryptography by removing the need of explicit public keys. The idea was to derive, in a publicly known way, a public key from an identity. Similarly, in secret-key cryptography, ID-based systems allow authorized entities to derive a secret key from an identity. So, large databases maintaining the correspondence between an identity and the corresponding secret key are no longer required, resulting in better performances and higher security.

Indexing terms: ID-based systems, Secret-key cryptography, Authentication protocols, One-time passwords, Kerberos, MACs

1 Introduction

Basically, there are two general forms of cryptography: *secret-key cryptography* (e.g., [1]) and *public-key cryptography* (e.g., [9, 18]). In secret-key cryptography, the security mainly rests on the (symmetric) key; someone in possession of the key can encrypt *and* decrypt messages. Public-key cryptography differs in the sense that everyone can encrypt messages while only authorized receivers can decrypt messages—the encryption and decryption keys are respectively called *public key* and *private key*.

Public-key cryptography presents the advantage that only the private key must be kept secret. It also greatly simplifies the key management. In secret-key cryptography, $(k^2 - k)/2$ keys are needed to allow a group of k users to privately communicate in pairs; moreover, these keys have to be transmitted via a secure channel. For large groups, the number of keys may become prohibitive and secure channels are not always available—a common remedy is to seek the

* This work was supported by the National Science Council of the Republic of China under contracts NSC87-2213-E-032-012 and NSC87-2811-E-032-0001.

help of a *Trusted Third Party* (TTP). When needed, the TTP generates 'on the fly' a session key for two users wishing to have a private communication. Note that in public-key cryptography, a TTP is also required to authenticate the public (enciphering/verification) keys. The main difference is that, in secret-key cryptography, the TTP must always be *unconditionally* trusted because it can easily impersonate a user.

In 1984, Shamir introduced *ID-based systems* [20] in order to avoid the explicit authentication of the public keys by means of *public-key certificates*. The idea was to create public-key cryptosystems wherein the identity of a user plays the role of his public key. From an user's identity (which is publicly known and in a standardized form), a TTP computes the corresponding private key and securely transmits it to the user. Here too, the TTP must be unconditionally trusted. Analogously, the idea behind *ID-based secret-key systems* is to derive the (secret) keys from the identity. Since, in secret-key cryptography, the security relies on these keys, this operation can only be performed by a TTP. This TTP also serves as an arbitrator when disputes arise due to a user denying certain actions.

One advantage of ID-based systems[1] over public-key systems is that public-key certificates are no longer necessary (and therefore do not have to be stored); a public key being implicitly certified from an identity. This possibly results in a saving of space requirements. Similarly, ID-based secret-key systems avoid a TTP to maintain a large database containing (information related to) the secret keys. This has the further advantage to offer a higher security level.

Last but not least, secret-key cryptosystems are several orders of magnitude faster and use keys that are generally smaller in comparison to public-key systems. ID-based secret-key systems provide thus a competitive alternative for medium-sized networks. The size restriction is due to the fact that the TTP may be involved in some interactions between users.

The rest of this paper is organized as follows. ID-based secret-key cryptography is defined in Section 2. In Section 3, several applications motivate the use of ID-based secret-key systems. Finally, Section 4 concludes the paper.

2 ID-based Secret-Key Cryptography

This section formally defines ID-based secret-key cryptography. The related notion of symmetric-key certificates is also discussed.

2.1 Basic functional model

In ID-based secret-key cryptography, the parties involved in any protocol are called *entities*; they can be users, groups of users, software programs, etc. Among entities U_i, one entity is privileged and trusted (think about the superuser in Un*x systems). This entity is called the *trusted authority* and is denoted as T.

[1] Unless explicitly mentioned, "ID-based system" has to be understood as ID-based *public-key* system, as originally defined by Shamir [20].

By abuse of language, *identity* refers to any public information that uniquely identify an entity (such as name and address, IP address, ...). The identity may also contain *attributes*, i.e. additional data that the application warrants. The identity of entity U_i is denoted ID_i. Note that, for each application, the format of the identity has to be publicly standardized; moreover to avoid simple forgeries, some redundancy rules like those defined in [2] can be added.

Each protocol supposes a set-up phase, which includes an initiation phase and a registration phase.

Initiation phase For a given set of applications and entities, the trusted authority owns a secret-key K, called the *master key*. Since the security of these applications depends on it, strong measures have to be taken to protect it (e.g., control procedures, tamper-resistant devices, ...).

Registration phase When entity U_i comes to the trusted authority, it is given an identity ID_i as defined before. From this identity (and optionally from private information h_i), the trusted authority computes U_i's secret key, denoted k_i, as follows

$$k_i = \mathrm{H}_K(ID_i, h_i^*), \tag{1}$$

where $\mathrm{H}_K(\cdot)$ is a one-way function keyed under the master key K. Examples of keyed one-way functions include DES [1], HMAC [14], ... The resulting key k_i is called *ID-key* of entity U_i. Note that, in Eq. (1), argument h_i is starred to indicate that it is optional.

For evident security reasons, ID-keys k_i have to be periodically updated. This may be achieved by including a lifetime attribute in the identity ID_i of entity U_i. This attribute may also serve to address the key revocation issue. It may also be prudent to include an application-dependent attribute; so if for any reason an ID-key is intercepted, it may only be used within the corresponding specific application. Alternatively, some attributes may be incorporated as private (or hidden) information into h_i (see Eq. (1)).

2.2 ID-keys vs. secret-key certificates

ID-based systems provide a means to avoid the explicit authentication of the public keys through public-key certificates. Similarly, the ID-keys k_i can be considered as implicit *symmetric-key certificates* [15, pp. 554–555], i.e. the symmetric encryption[2] of an entity's secret-key under the master key of a trusted authority. Example 1 shows how secret-key certificates allow an entity U_1 to transmit a secret message to another entity U_2; the main advantage being that the trusted authority stores in a database the secret-key certificates rather than the secret keys. Further discussion (under the name "private-key certificates") on their use can be found in [7].

[2] Our definition (see Eq. (1)) is more general; the keyed one-way function H_K is not restricted to symmetric encryption algorithms.

Example 1 (Transmission of a secret message from U_1 to U_2). Let $\mathrm{E}_k(\cdot)$ be a symmetric encryption algorithm parametrized by a secret-key k. Let K be the master key of a trusted authority T and let k_i be the secret key of entity U_i. The secret-key certificate of U_i is $Cert_i = \mathrm{E}_K(k_i)$. Then the protocol goes as follows. U_1 computes $c = \mathrm{E}_{k_1}(U_2, m)$ and sends it to T. Upon looking up the database, T computes $k_1 = \mathrm{E}_K(Cert_1)$ and hence decrypts c to see that the message is for U_2. T looks up again the database to calculate U_2's secret key $k_2 = \mathrm{E}_K(Cert_2)$. Then T computes $c' = \mathrm{E}_{k_2}(m, U_1)$ and sends it to U_2. From c', U_2 recovers the message m (along with the identity of U_1). Schematically,

$$\begin{array}{ll} \text{(S1)} & U_1 \rightarrow T \ : \ U_1, \mathrm{E}_{k_1}(U_2, m) \\ \text{(S2)} & T \rightarrow U_2 : \ \mathrm{E}_{k_2}(m, U_1) \end{array}$$

Alternatively, to avoid T to look up the database, U_1 may send $Cert_1$ and $Cert_2$ in Step (S1). However, in that case, the database must be public. □

Our model differs from symmetric-key certificates in that a database for storing the certificates is no longer required. Another approach to avoid the requirement of maintaining this database is to ask each entity to provide the trusted authority with his (her) own certificate when needed. So, the trusted authority has not to store the certificates. This is illustrated in the following example.

Example 2 (Transmission of a secret message to T). The notations are the same as in Example 1. Suppose that U_1 wants to transfer a secret message m to the trusted authority T. Then U_1 encrypts m as $c = E_{k_1}(m)$ and sends $(c, Cert_1)$ to T. From $Cert_1$, T recovers $k_1 = \mathrm{E}_K(Cert_1)$ and therefore $m = E_{k_1}(c)$. Schematically,

$$\text{(T1)} \qquad U_1 \rightarrow T : \ U_1, \mathrm{E}_{k_1}(m), Cert_1$$ □

The major drawback of the above approach is that *translation requests* are impossible. Since U_i and only U_i knows $Cert_i$, it is for instance impossible for entity U_1 to transmit a secret message to another entity U_2 as in Example 1. Indeed, the trusted authority T is unable to compute U_2's secret key k_2 because he does not know the corresponding certificate $Cert_2 = \mathrm{E}_K(k_2)$.

ID-based secret-key cryptography does not suffer from such limitations. Translation requests are possible *without* database containing entity's secret-key certificates. As it will be shown in the next section, such a database may be very sensitive to dictionary attacks.

3 Applications

In public-key cryptography, ID-based systems find natural applications in interactive identification and signature protocols such as those proposed by Fiat and Shamir [11]. This section gives their counterpart in the secret-key setting. The presented protocols are intentionally simple. However, they can easily be

customized to offer additional functionalities or serve as basic tools to construct more sophisticated services. Our objective is to emphasize the advantages of ID-based secret-key systems over usual secret-key systems.

3.1 Login protocols for computer networks

Traditional approaches for login protocols can no longer be adequate for modern usage, especially over a distributed network. Once the password of a legitimate entity U has been intercepted, the eavesdropper can impersonate U by replaying the password of U. Resistance against replay attack can be obtained through the use of *one-time passwords*. A nonce (i.e., timestamp, sequential or challenge number) is encrypted using a conventional secret-key cryptosystem, and then is transmitted as evidence of identity proof [3]. Alternatively, the secret-key encryption function may be replaced by a keyed hash function [4].

Another weakness of login protocols resides in the privacy of the password file. So, instead of storing passwords, system administrators usually store one-way functions of passwords in a write-protected file [10, 16]. Therefore even if someone breaks into the computer and steals the password file, he cannot recover the passwords because of the one-way function applied on the passwords. Unfortunately, the use of one-way functions only partially solve the problem: the password file is still vulnerable to dictionary attacks. For example, the `crack` program reveals itself to be very effective in practice [12]. For that reason, most modern operating systems propose now in standard to store the passwords in a *shadowed* file, that is, a *read-* and write-protected file by the system administrator.[3] This is a major improvement over previous solutions because it disallows simple off-line password guessings. ID-based secret-key cryptography pushes this idea further by completely removing the need of a password file.

Timestamp-based authentication Suppose that entity U_i wants to authenticate itself to T. When logging to the system, U_i gives the *one-time password* $\mathrm{E}_{k_i}(time)$ along with its identity ID_i, where $\mathrm{E}_{k_i}(\cdot)$ is a symmetric encryption algorithm (e.g., [1]), k_i is U_i's ID-key and $time$ is the current time accessed from U_i's machine. (See Fig. 1.)

$$U_i \to T: \quad \mathrm{E}_{k_i}(time), ID_i$$

Fig. 1. Timestamp-based authentication.

Upon receiving $(\mathrm{E}_{k_i}(time), ID_i)$, computer T reconstructs U_i's ID-key k_i from the master key K and ID_i. It then decrypts $\mathrm{E}_{k_i}(time)$ to obtain the timestamp $time$. If $time$ is within the allowable range for both clock skew and transmission delay, the one-time password $\mathrm{E}_{k_i}(time)$ is accepted; otherwise it is rejected.

[3] What we call "trusted authority" in our model.

Challenge-response authentication In any timestamp-based authentication scheme, in order to avoid possible *short-term* replay attacks, either the clocks of the prover and the (remote) system must get synchronized or a table of previously accepted while unexpired passwords should be maintained in the (remote) system [5, p. 120] when loosely synchronized clocks (e.g., within one minute clock skew) are used. Each received password should be exclusive of this table. Evidently, the table should be *write-protected* for general users. Maintaining the table of course brings some spatial and computational complexities depending on the implementation. It also weakens the system.

$$\begin{aligned} T \to U_i &: \quad r \\ U_i \to T &: \quad \mathrm{E}_{k_i}(r), ID_i \end{aligned}$$

Fig. 2. Challenge-response authentication.

A possible solution is to replace the timestamp by a challenge-response nonce at the cost of one additional interaction between the prover and the verifier (see Fig. 2). The verifier T selects a random number r and sends it to prover U_i. Then, U_i responds $(\mathrm{E}_{k_i}(r), ID_i)$.

As mentioned before, the symmetric-encryption algorithm $\mathrm{E}_{k_i}(\cdot)$ may be replaced by a one-way hash function (e.g., [14]). This may be a political advantage when an encryption algorithm is subject to export restrictions.

ID-based secret-key cryptography not only has applications in simple authentication protocols, but also in more sophisticated applications such as Kerberos.

Four-headed dog: enhancing Kerberos In 1989 began the design of Kerberos[4] [13]: an authentication protocol for TCP/IP networks. This protocol allows an entity U_1 to use a service offered by another entity U_2. It can synthetically be described as follows. Entity U_1 first requests a "ticket" from a trusted third-party T. Then U_1 presents this ticket to authenticate itself to U_2. For this purpose, the trusted third-party (Kerberos server) owns a database containing the name, the secret key, the expiration time of the secret key and some administrative information of each entity. Since the security of the system relies on this database, it is assumed the Kerberos runs on a *secure* server. This assumption may be hard to be satisfied in practice. For example, entities may use the same passwords on different systems; or the database may be found on a stolen backup. ID-based secret-key cryptography does not require such working assumptions, and therefore enhances the overall security of Kerberos.

[4] Kerberos is a mythological three-headed dog guarding Hades.

3.2 ID-based message authentication code

The challenge-response authentication protocol (see Fig. 2) can be turned into a "signature" protocol by replacing the challenge r by the message to be signed. In the secret-key language, a signature is usually called a *message authentication code* (MAC). The difference is that the "verification" key is secret. Similarly to a regular signature, the purpose of a MAC is to authenticate *both* the source and the integrity of a message.

$$U_i \rightarrow T: \quad m, \mathrm{E}_{k_i}(m), ID_i$$

Fig. 3. ID-based MAC algorithm.

Upon receiving $(m, \mathrm{E}_{k_i}(m), ID_i)$ from entity U_i, the trusted authority T computes $k'_i = \mathrm{H}_K(ID_i)$ and then checks whether or not $\mathrm{E}_{k'_i}(m) = \mathrm{E}_{k_i}(m)$. If these two values match then T is assured that message m has not been modified and that it effectively comes from U_i because nobody else can produce $\mathrm{E}_{k_i}(m)$.

4 Conclusions

ID-based secret-key cryptography where secret keys are *privately* binded to entity's identities was proposed. Although related to the notion of secret-key certificates, ID-based secret-key systems present the further advantage to allow translation requests *without* any database maintaining the correspondence between the identity of each entity and its secret-key certificate. This results in a simultaneous reduction of both transmission bandwidth and storage requirements. More important, this increases the security. On the other hand, ID-based secret-key systems offer superior computational performances to that of their public-key analogues. They thus provide a competitive solution for medium-sized networks.

References

1. FIPS PUB 46. Data encryption standard. Federal Information Processing Standards Publication, U.S. Department of Commerce, National Bureau of Standards, Springfield (Virginia), 1977.
2. ISO/IEC 9796. Information technology – Security techniques – Digital signature scheme giving message recovery. International Organization for Standardization, Geneva (Switzerland), 1991.
3. ISO/IEC 9798-2. Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms. International Organization for Standardization, Geneva (Switzerland), 1994.
4. ISO/IEC 9798-4. Information technology – Security techniques – Entity authentication – Part 4: Mechanisms using a cryptographic check function. International Organization for Standardization, Geneva (Switzerland), 1995.

5. W.R. Cheswick and S.M. Bellovin. *Firewalls and Internet security.* Addison-Wesley, 1994.
6. D.W. Davies and W.L. Price. *Security for computer networks.* John Wiley & Sons, 2nd edition, 1989.
7. D. Davis and R. Swick. Network security via private-key certificates. *Operating Systems Review*, 24:64–67, 1990.
8. D. de Waleffe and J.-J. Quisquater. Better login protocols for computer networks. In B. Preneel, R. Govaerts, and J. Vandewalle, editors, *Computer Security and Industrial Cryptography*, volume 741 of *Lecture Notes in Computer Science*, pages 50–70. Springer-Verlag, 1993.
9. W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
10. A. Evans Jr, W. Kantrowitz, and E. Weiss. A user identification scheme not requiring secrecy in the computer. *Communications of the ACM*, 17(8):437–442, August 1974.
11. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.
12. D.V. Klein. Foiling the cracker: a survey of, and improvements to password security. In *Proc. of the 2nd USENIX Security Workshop*, pages 5–14, 1990.
13. J. Kohl and C. Neuman. The Kerberos network authentication service (V5). Internet Request for Comments RFC 1510, September 1993. Available at `ftp://ds.internic.net/rfc/rfc1510.txt`.
14. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: keyed-hashing for message authentication. Internet Request for Comments RFC 2104, February 1997. Available at `ftp://ds.internic.net/rfc/rfc2104.txt`.
15. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997.
16. G.P. Purdy. A high security log-in procedure. *Communications of the ACM*, 17(8):442–445, August 1974.
17. R. Rivest. The MD5 message digest algorithm. Internet Request for Comments RFC 1321, April 1992. Available at `ftp://ds.internic.net/rfc/rfc1321.txt`.
18. R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
19. B. Schneier. *Applied cryptography: Protocols, algorithms, and source code in C.* John Wiley & Sons, 2nd edition, 1996.
20. A. Shamir. Identity-based cryptosystems and signature schemes. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology — Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.
21. G.J. Simmons, editor. *Contemporary cryptology: The science of information integrity.* IEEE Press, 1992.