CITI Technical Report 98–4

Implementation of a Provably Secure, Smartcard-based Key Distribution Protocol

Rob Jerdonek
Peter Honeyman
Kevin Coffman
Jim Rees
Kip Wheeler[1]

ABSTRACT

We describe the implementation of the Shoup-Rubin key distribution protocol. This protocol stores long-term keys on a smartcard and uses the cryptographic capability of the card to generate and distribute session keys securely. The designers of the protocol provide a mathematical proof of its security, using techniques pioneered by Bellare and Rogaway. Combining this theoretical strength with our implementation on tamper resistant hardware results in practical and powerful middleware functionality, useful in applications that demand strong authentication and confidentiality.

June 24, 1998

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

Implementation of a Provably Secure, Smartcard-based Key Distribution Protocol

Rob Jerdonek
Peter Honeyman
Kevin Coffman
Jim Rees
Kip Wheeler[1]

1. Introduction

The Center for Information Technology Integration is a research laboratory that addresses near- and intermediate-term challenges to the University of Michigan's information technology environment. It does this by establishing collaborative relationships with industrial partners to explore and develop enterprise-scale information technology solutions. From the start, CITI's focus has been on middleware. In recent years, increasing amounts of attention have been focused on computer and network security.

Early in 1996, CITI began exploring ways to implement special functionality in smartcards to improve secure access to distributed services. The most prominent flaw in the University of Michigan computing environment (UMCE) is its reliance on the strength of user-selected passwords. UMCE makes heavy use of Kerberos IV for many secure services, such as login authentication, filing, email, and web access; regrettably, Kerberos IV admits an offline dictionary attack that is difficult to detect and defeat. Such an attack is likely to succeed any time users are responsible for selecting their own passwords or pass phrases [1]; indeed, it has long troubled CITI to confirm that many weak passwords are in daily use in the UMCE.[2]

Our goal is to replace password-based authentication with stronger means. Smartcards bear particular attraction, as they are able to store securely moderate amounts of information, such as cryptographic keys; offer good tamper-resistance; and can even perform cryptographic calculations with modest performance.

Establishment of a session key is central to the goal of mutual authentication of cooperating principals in a security domain. Principals establish a security context by agreeing on a shared secret, which is used to authenticate or secure subsequent communications. An impractical approach is to arrange that all principals in a security domain share mutual secret keys; this arrangement breaks down from quadratic growth in the number of keys that must be set up in advance and the concomitant requirement that principals manage private databases of keys. Needham and Schroeder observe that a trusted third party can reduce these complexities by sharing a long-term key with each of the principals in the security domain [2]. This has two distinct advantages. First, O(n) long-term keys are needed, instead of $O(n^2)$. Second, each principal must maintain only the key that it shares with the third-party, rather than one key for each of the other principals in the security domain.

While this reduces the obligations and bookkeeping for principals, it does not eliminate their responsibilities altogether, nor shield them from disaster in the event that control over the long-term key is lost. To assist principals in the secure management of their keys, Bellcore researchers Victor Shoup and Avi Rubin devised an innovative key distribution protocol that exploits the tamper-resistant properties of smartcards to provide a convenient and secure repository for cryptographic keys [3]. With the help of Personal Cipher Card Corporation (PC^3), CITI implemented the Shoup-Rubin protocol on smartcards and host workstations.

1 Personal Cipher Card Corporation, Lakeland, FL, `smartcard1@compuserve.com`

2 Although it is somewhat soothing to discover that the most common password is *love*.

Shoup-Rubin uses two types of cryptographic keys. Long-term keys are securely stored on the card, never leaving its physical boundaries. These keys are shared with a trusted third-party and used to establish (short-term) session keys. Session keys are not (necessarily) stored on secure, tamper-resistant hardware, so they are vulnerable to compromise. In contrast, long-term keys must never be vulnerable.

The details of Shoup-Rubin, outlined in the next section, are fairly intricate, in part to satisfy the requirements of an underlying complexity-theoretic framework [4]. This inconvenience is balanced by the ability to prove powerful properties of the protocol. In particular, Shoup and Rubin are able to prove that the protocol does not disclose the session key to an adversary. The combined strength of mathematical proof of security and a tamper-resistant implementation lends good confidence in the overall security of our approach.

2. Shoup-Rubin Key Distribution Protocol

The Shoup-Rubin key distribution protocol runs among three parties: the communicating peers, and a third-party that is trusted to hold long-term keys securely. Following Schneier [5], we call them ALICE, BOB, and TRENT, respectively. ALICE and BOB can each be viewed as a pair of agents, one responsible for holding long-term keys, and one with less stringent security requirements. The former agent is implemented on a secure token such as a smartcard, while the latter runs on an ordinary computer. We rely on the tamper-resistance inherent in smartcards to protect the long-term keys used by Shoup-Rubin. The host computers have less stringent security requirements; they rely on physical and other security properties inherent to the secure tokens holding the long-term keys.

We now describe the Shoup-Rubin session key distribution protocol, first in general terms, then in detail. We identify ALICE and BOB's computers with ALICE and BOB themselves. We assume that ALICE's and BOB's computers have smartcard readers attached, and that no computer other than the one to which it is attached can communicate with a smartcard reader. Shoup-Rubin builds on the Leighton-Micali key distribution protocol [6], an inexpensive symmetric key distribution protocol. Leighton-Micali uses a construct known as a *pair key* to establish a shared secret between communicating parties.

Let **A** and **B** denote unambiguous identifiers for ALICE and BOB, and let K_A and K_B be their long term keys. These keys are shared with TRENT. DES encryption of message M with key K is denoted $\{M\}_K$. ALICE and BOB's *pair key* is defined

$$\Pi_{AB} = \{\mathbf{A}\}_{K_B} \oplus \{\mathbf{B}\}_{K_A}$$

TRENT calculates pair keys on demand; that is TRENT's entire role. Because a pair key reveals nothing about the long-term keys used in its calculation, it can be communicated in the clear.

With pair key Π_{AB} in hand, ALICE computes $\{\mathbf{B}\}_{K_A}$. Combining this with the pair key yields $\kappa = \{\mathbf{A}\}_{K_B}$. BOB can compute κ directly, so once ALICE has a pair key in hand, she and BOB can communicate privately using key κ.

Shoup-Rubin extends Leighton-Micali in two ways:

- κ is computed on ALICE's and BOB's smartcards, freeing ALICE and BOB from the responsibility of knowing their own long-term keys.
- ALICE and BOB then use κ to secure the messages that provide for session key agreement.

Shoup and Rubin use Bellare and Rogaway's innovative complexity theoretic techniques [4] to prove that their key distribution algorithm does not disclose the session key, even to an extremely powerful adversary. We now describe the Shoup-Rubin protocol in detail.

2.1. Details

The following table defines the terms used in the Shoup-Rubin smartcard-based session key distribution protocol. Integer operands are concatenated to other protocol terms with the ‘‘dot’’ operator to satisfy requirements of the Bellare-Rogaway proof framework.

Term	Meaning
A, **B**	Unique identifiers
K_A, K_B	Long-term keys
K_{AC}, K_{BC}	Secret card keys
r, s	Nonces
$\Pi_{AB} = \{\mathbf{A}\cdot 0\}_{K_B} \oplus \{\mathbf{B}\cdot 1\}_{K_A}$	Pair key
$\alpha = \{\Pi_{AB}\cdot\mathbf{B}\cdot 2\}_{K_A}$	Verifies Π_{AB}
$\beta = \{r\cdot s\cdot 1\}_{\kappa}$	Verifies r and s
$\gamma = \{r\cdot 1\cdot 1\}_{K_{AC}}$	Verifies r
$\delta = \{s\cdot 0\cdot 1\}_{\kappa}$	Verifies s
$\kappa = \{\mathbf{A}\cdot 0\}_{K_B}$	See discussion
$\sigma = \{s\cdot 0\cdot 0\}_{\kappa}$	Session key

Shoup-Rubin glossary

We now detail the steps of Shoup-Rubin.

From	To	Message	Meaning
ALICE	TRENT	**A**, **B**	ALICE wishes to initiate a session with BOB.
TRENT	ALICE	Π_{AB}, α	Π_{AB} is ALICE and BOB's pair key. α is a verifier for Π_{AB}.

ALICE asks TRENT for the ALICE/BOB pair key. TRENT also returns a verifier, which ALICE's card uses to prevent masquerading.

From	To	Message	Meaning
ALICE	CardA	—	ALICE requests a nonce to verify subsequent communication with BOB.
CardA	ALICE	r, γ	r is a nonce, γ is a verifier for r.

ALICE's first card operation

ALICE initiates the protocol with BOB by requesting a nonce from her smartcard. ALICE retains the verifier for later use.

From	To	Message	Meaning
ALICE	BOB	**A**, r	BOB will use r to assure ALICE of his correct behavior.

By sending a nonce to BOB, ALICE requests establishment of a fresh session key.

From	To	Message	Meaning
BOB	CardB	**A**, r	BOB instructs his smartcard to construct a session key, and provides ALICE's nonce for her subsequent verification.
CardB	BOB	s, σ, β, δ	s is a nonce used to construct the session key. σ is the session key. β is ALICE's verifier for r and s. δ is BOB's verifier for s.

BOB's card operation

BOB sends ALICE's identity and her nonce to his smartcard. BOB's card calculates κ, then generates a nonce and a session key. BOB's card also generates two verifiers; one is used by ALICE's card to verify both nonces, the other is used by BOB to verify ALICE's subsequent acknowledgement. BOB retains the session key and his verifier.

From	To	Message	Meaning
BOB	ALICE	s, β	ALICE needs s to construct the session key, and β to verify r and s. BOB retains σ, the session key, and δ, a verifier for s.

BOB forwards his nonce, from which ALICE's card constructs the session key.

From	To	Message	Meaning
ALICE	CardA	**B**, r, s, Π_{AB}, α, β, γ	Verify: Π_{AB} with α, r with γ, and BOB's use of r and s with β. Use Π_{AB} and s to construct the session key.
CardA	ALICE	σ, δ	σ is the session key. δ is sent to BOB to confirm ALICE's verification of s.

ALICE's second card operation

ALICE sends everything she has to her smartcard: BOB's identity, the pair key and its verifier, her nonce and its verifier, and BOB's nonce and its verifier. ALICE's card validates all the verifiers. If everything checks out, ALICE's smartcard derives κ, then constructs the session key from BOB's nonce and uploads it to ALICE along with a verifier to assure BOB that ALICE is behaving properly.

From	To	Message	Meaning
ALICE	BOB	δ	Confirm

ALICE sends the verifier to BOB. BOB compares it to his retained verifier.

2.2. Implementation

To implement Shoup-Rubin, CITI turned to PC^3. PC^3 extended SCOS, a proprietary cryptographic smartcard operating system, to support four new operations, $KeyDist_1$, $KeyDist_2$, $KeyDist_{3a}$, and $KeyDist_{3b}$, and supplied CITI with SCOS cards that implement these operations. CITI tested the cards extensively, both for correctness and for performance. Performance is of interest because it affects usability of the implementation. If card operations take too many seconds, the user experience is adversely affected.

To gauge performance, CITI built SCOS drivers for UNIX, Windows 95, and NT that record communication time and elapsed time. The difference between elapsed time and I/O time is accounted to card processing time. For Shoup-Rubin operations that require more than one SCOS command, we also measure host processing time, but this is negligible in all cases.[3] Although this paper reports only UNIX performance, NT performance is comparable. We encountered timer granularity problems on Windows 95 that affected performance adversely.

ALICE's first card operation is implemented directly by $KeyDist_1$. This requires two I/O operations: one to send an SCOS command string, and one to read results from the card.

SCOS does not send both values and results in a single command, so BOB's card operation is broken into two SCOS commands: one to send the command and values, and one to read the results from an output buffer.

ALICE's second card operation also has values and results, requiring multiple SCOS commands. SCOS can send up to 64 bytes in one command, but this operation sends 72 bytes to the card, so we use three SCOS operations to effect $KeyDist_3$: $KeyDist_{3a}$ sends the first 64 bytes of values and $KeyDist_{3b}$ sends the remaining 8 bytes. Then an SCOS read command recovers the results from an output buffer.

[3] < 1 msec

2.3. Performance

PC^3's implementation of Shoup-Rubin runs on the SGS-Thomson ST16F48 card, which is based on an MC68HC05 microprocessor clocked at 3.57 Mhz, with 8 KB EEPROM, 16 KB ROM, and 384 bytes RAM. Version PC64T4 of SCOS uses 128 bytes of EEPROM, 6K of ROM, and 128 bytes of RAM. The Shoup-Rubin extensions to SCOS are written in 250 lines of Motorola 6805 assembler. This assembles to 430 bytes of code, which is stored in an executable EEPROM file.

ALICE runs on a 200 MHz Pentium running OpenBSD 2.2 and BOB runs on a 122 Mhz PowerPC running AIX 4.2. Serial communication with the card is 9600 bps.

The following table summarizes the number of bytes transmitted, and host, card, and communication times for the three card operations. Each figure represents the average of 10 time trials. Variance among the trials is negligible. All times are in msec.

operation	cmds	bytes	comm	card	clock
$KeyDist_1$	1	32	32	63	95
$KeyDist_2$	2	80	306	147	454
$KeyDist_3$	3	112	133	241	374

Total time from the start of $KeyDist_1$ to the completion of $KeyDist_3$ is 3.1 sec. Most of the time is spent in network communications and protocol processing.

Our UNIX drivers write a byte at a time. Evidently OpenBSD handles this well, averaging slightly more than the expected msec per byte, while AIX is apparently tuned for larger transfers.

3. Discussion

To demonstrate the capabilities of our smartcard-based Shoup-Rubin implementation, we wrapped it in a GSS API interface library [7] on which we built a secure videoconferencing application [8]. While implementing in SCOS was a successful strategy, an approach that gives CITI more direct control over card programming would be more satisfactory.[4] But even at this writing, it is not easy for University developers to obtain cards and documentation for programmable smartcards capable of general-purpose cryptographic applications. For example, first generation JavaCards lack cryptography.

[4] To CITI.

Yet more frustrating, on several occasions, CITI has been offered programmable cryptographic smartcards with embedded DES capability, only to find the DES engine ‘‘crippled’’ by key length or other limitations intended to satisfy US export regulations. (More than once, CITI has pled with manufacturers to consult an atlas.) These difficulties aside, this is an exciting time to be working with secure tokens: new companies and products are making custom programming of secure tokens fast and easy with user-friendly development kits that support high-level languages and rapid prototyping.

Acknowledgements

We thank Brahm Windeler for help in collecting the timings. This work was partially supported by a grant from Bellcore.

References

1. Robert Morris and Ken Thompson, ‘‘Password Security: A Case History,’’ *Communications of the ACM* **22**(11) (November, 1979).
2. R.M. Needham and M.D. Schroeder, ‘‘Using Encryption for Authentication in Large Networks of Computers,’’ *Communications of the ACM* **21**(12) (December, 1978).
3. V. Shoup and A.D. Rubin, ‘‘Session Key Distribution Using Smart Cards,’’ in *Proc. of Eurocrypt ’96* (May, 1996).
4. M. Bellare and P. Rogaway, ‘‘Provably Secure Session Key Distribution: The Three Party Case,’’ in *Proc. ACM 27th Ann. Symp. on the Theory of Computing* (1995).
5. B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc. (1996).
6. T. Leighton and S. Micali, ‘‘Secret-Key Agreement Without Public-Key Cryptography,’’ pp. 456–479 in *Proc of Crypto ’93*, Santa Barbara (1993).
7. J. Linn, ‘‘Generic Security Service Application Program Interface, Version 2,’’ RFC 2078 , USC/Information Sciences Institute (January, 10, 1997).
8. Peter Honeyman, Andy Adamson, Kevin Coffman, Janani Janakiraman, Rob Jerdonek, and Jim Rees, ‘‘Secure Videoconferencing,’’ pp. 123–130 in *Proc. 7th USENIX Security Symp.*, San Antonio (January, 1998).