A PROPOSAL OF ARCHITECTURAL ELEMENTS FOR IMPLEMENTING SECURE SOFTWARE DOWNLOAD SERVICE IN SOFTWARE DEFINED RADIO

Lachlan B. Michael*, Miodrag J. Mihaljević*†, Shinichiro Haruyama* and Ryuji Kohno‡

*Sony Computer Science Laboratories, Inc., Tokyo, Japan, michael@csl.sony.co.jp
†Math. Inst., Serbian Academy of Sciences and Arts, Belgrade, Yugoslavia, miodragm@turing.mi.sanu.ac.yu
‡Yokohama National University, Japan.

Abstract - In order to obtain an appropriate, high level of security, a number of architectural elements for secure downloading of software to a software defined radio (SDR) terminal have been pointed out. They include four different cryptographic techniques and employment of tamper resistant hardware. The cryptographic techniques employed are: (a) a secret key encryption technique; (b) a public key encryption technique; (c) a technique for cryptographic hashing and (d) a technique for digital signature. Particularly, a protocol for exchanging cryptographic components in an automatic manner without any assistance from the user, is proposed. Implementation characteristics of certain cryptographic components are also discussed.

Keywords - software defined radio, software download, reconfigurable logic, security, cryptography

I. INTRODUCTION

One of the most pressing issues for the commercial introduction of software defined radio (SDR) systems ([1] - [5]) is the authentification and verification of integrity of the software that is downloaded [6]. For a SDR terminal, since reprogrammable hardware is used, if the software is illegally modified from when it was submitted to the authorities, then the use of such software may cause the wireless device to emit radiation illegally, which may cause interference to other users or physical harm to the user.

Therefore, there must be a method of ensuring that the software downloaded is intact and has not been modified (verification of integrity) and that it has obtained government approval (authentification). Furthermore, in the event that some illegally modified software is created, there should be some mechanism to prevent the spread of that illegal software. As a further necessity for the introduction of a software downloadable SDR system, the software should be protected against theft by people or companies who would like to know the details of the software employed by a rival company.

Although the significance of the security issue is recognized in a number of documents and papers, precise proposal of a framework for solving the problem has been considered only very recently.

In this paper, following the recently reported results in [7], we discuss certain architectural elements of a system for secure software download. The consideration includes a proposal for the protocol framework for exchanging cryptographic components, as well as discussion of certain implementation issues.

Section II summarizes the background and motivation for our work. A system for the secure downloading is summarized in Section III. An approach for changing the cryptographic components is proposed in Section IV. Certain implementation issues are discussed in Section V. Finally, the main conclusions are pointed out in Section VI.

II. BACKGROUND AND MOTIVATION

As a straightforward attempt based on existing security techniques developed for Internet or wireless communications, several methods of secure download have been considered ([9], [10]) or could be considered. These include SSL based security, WEP (Wired Equivalent Privacy) based security and some dedicated SDR proposals. The European group IST-TRUST has been undertaking research into secure software download ([3], [4], [11]).

On the other hand it is important to note that recently it has been shown that both SSL and WEP have certain weaknesses ([12], [13], [14]). The discussion of SSL and WEP implies that current versions cannot be recommended for SDR purposes.

In [7] a framework for the secure download of software, including the ability to exchange security components was proposed. This framework yields a conceptual solution and its intention was not to discuss all details, particularly due to the fact that details of software downloading itself are not specified yet [8], and so development of these details is required. Accordingly, the intention of this paper is to address some of these issues.

III. A FRAMEWORK FOR THE SECURE DOWNLOAD

A. Main Characteristics

The proposed secure downloading system is based on certain collections of cryptographic primitives (hashing, digital signature and ciphering) and the keys. During downloading, only one component selected from each of the collections is employed.

In the proposed system a unique collection of secret keys

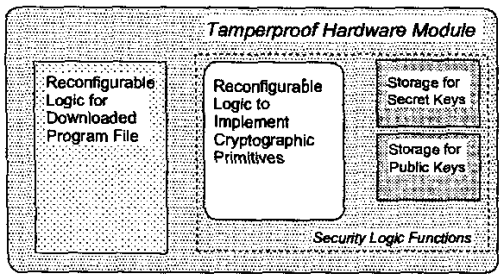

Fig. 1. Tamperproof security module showing four major components.

is assigned for the purpose of symmetric encryption to each wireless terminal. Tamper resistant hardware is employed to provide a secure storage for the terminal secret key to be used for decrypting of the symmetric key encrypted data (Fig.1). It has been reported that software implementation of security functions offers only limited physical security, particularly with respect to the key storage [18]. Each terminal also contains a collection of public keys of the verification or approval authority (usually the government) for the data to be downloaded.

We note that the proposed system is an end-to-end encryption system, that is, the data is encrypted at (or very close to) the source and decrypted at the wireless terminal rather than a wireless security system. Most current proposals are wireless standards security proposals. That is, the security is employed by the wireless provider alone, for the over-the-air link, and this has two drawbacks. First, the attacker could try and steal the software before being transmitted to the user and secondly and more importantly, all users and terminals must use the same security measures. In the case that a weakness exists, the standards must be revised, as has been shown with security in WEP or GSM, for example. Furthermore the wireless link may not be the only method for downloading of software.

Software defined radios will encompass several wireless standards. Therefore, a system is required that is independent of any particular wireless standard used. The proposed system is independent of any wireless security methods employed.

B. Digital Signature for Authentification of Origin and Verification of Integrity

The generation of the digital signature is shown in Fig. 2.

First, the software or program file and the appropriate hardware is submitted to the authorities for testing, or as is currently the case test results are submitted and approved. Once the combination of software and hardware has been approved, the fingerprint or hash H of the software will be signed using the approval authorities secret key SK. No one apart from the appropriate agency has access to or knowledge of this secret key. The digital signature DS generated is unique for a particular piece of software D due to the nature of the hash

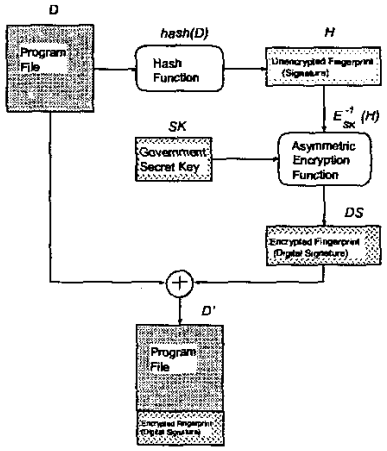

Fig. 2. Digital signing of program file by government approval agency after testing and ensuring software meets appropriate regulatory guidelines.

function $hash$, and now if the terminal manufacturer changes the software, they must reapply to have the software validated again.

The digital signature is

$$DS = E_{SK}^{-1}(H) \quad \text{where} \quad H = hash(D)\,.$$

The digital signature consists of a hash and an encryption function. Even if the hash function is a well known one, the encryption of the hash function by the approval authorities secret key SK makes it extremely difficult to forge the digital signature for that piece of software.The software program file is now signed as being legitimate, that is approved software. However, the program file is still able to be read and is not encrypted.

C. Encryption to a Particular Terminal

The encryption of the file and the digital signature is done by using a secret key K, which is unique to each terminal. One copy of the key is stored in the tamperpr only that terminal has the knowledge of the secret key. The secret key is stored in tamper proof hardware on the terminal device. Since symmetric encryption techniques are used, the encryption and decryption is usually faster then asymmetric techniques. This is an advantage for real-time encryption and also for speedy decryption of the encrypted file. Savings in battery consumption can also be expected compared to asymmetric encryption techniques.

In the unlikely event that a secret key is stolen, the attacker still cannot generate legitimate software since they cannot correctly digitally sign any software without also the government's secret key. Even if that was stolen as well, the most danger that could be done is the generation of illegal software for at most one terminal. Incorrect or otherwise modified software cannot be distributed to every terminal without knowledge of each terminals' individual secret key.

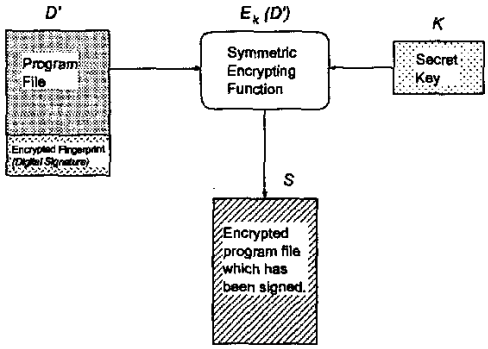

Fig. 3. Generation of encrypted program file to a specific terminal by the handset manufacturer by using the stored secret key of that particular terminal.

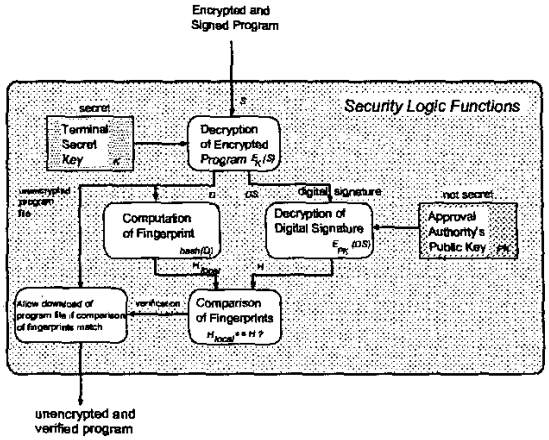

Fig. 4. Processing of the downloaded program file on the terminal hardware. Specifically, the security logic component is shown here.

It must be noted that since the distribution of keys is accomplished at the time of manufacture, there is no need to have a method to securely distribute the keys after that time.

D. Operation at Software Defined Radio

The decryption and verification of the downloaded software is shown in Fig. 4. First the encrypted file is decrypted using the terminal secret key K.

$$D' = E_K(S) \text{ where } D' = D + DS$$

Next, the digital signature DS is checked using the approval authorities' public key PK, available to all terminals.

$$H = E_{PK}(DS)$$

Using the hash function the decrypted file hash or fingerprint is calculated $H_{LOCAL} = hash(D)$, and if the two match $(H == H_{LOCAL})$ then the software is legitimate and has not been modified since it was approved.

Based on this verification of integrity and authentification, the file should be downloaded into the reconfigurable logic, such as an FPGA. If the fingerprints do not match, then the software has been modified or is not signed and approved by the government, and is not loaded and the appropriate error messages should be displayed to the user.

IV. A PROTOCOL FOR EXCHANGING OF CRYPTOGRAPHIC COMPONENTS

Our proposal includes the possibility to change any of the cryptographic components employed. Inclusion of this possibility was motivated by the reality that the security evaluation of currently available cryptographic techniques can point out the weaknesses but can not yield a definitive proof of security. In other words, the current security evaluation cannot identify all weaknesses of a cryptographic primitive. It may be broken by some cryptanalysis techniques developed at a later stage. A recent discussion of this issue is reported in [16], as well. An illustrative example is the A5 cipher used in GSM. After a number of years and widespread use serious weaknesses and its breakability have been reported [15].

In order to minimize the negative effect of the possible situation that a manufacturer initially has employed a cryptographic primitive which appears to be undesirably weak, or a key is stolen or compromised, the proposal of the system for secure downloading includes the possibility for exchanging any of the currently used cryptographic primitives or keys with a new one, in an automatic manner.

A. Underlying Ideas

The goals of the proposal for the changeability of the cryptographic components include that the exchange should be completely automatic, requiring no intervention by the user. The user should not be aware that an exchange has been performed.

In order to support changeability of the cryptographic components we propose an approach which includes:

- a collection of at least two different cryptographic primitives for each of the required cryptographic functions (hashing, digital signature, encryption); each collection contains an element which is used by default, and a number of optional ones (with at least one optional element);
- a collection of the keys for the symmetric ciphering and digital signature, each containing at least two different keys;
- a protocol for updating the collections of the cryptographic primitives and the keys.

The addition of redundancy in each cryptographic component is an extra burden for the SDR terminal, however combined with an appropriate protocol it provides security even when some components are compromised. It is assumed that in each time instant at least one element from each of the collections of cryptographic components can be considered secure and that the software downloading protocol should

support downloading of an alternative cryptographic components. The procedure for updating a certain element of each of the collections in general is carried out at the same time as other software is downloaded (see Fig.5(b)).

The downloading protocol includes information whether the default cryptographic primitives and keys should be employed or other selections should be made. For example encryption primitive #1 is used by default, but a weakness is discovered. For the next program file software download (and for the download of new software to replace encryption primitive #1) we need the download protocol to specify that encryption primitive #2 should be used until #1 has been replaced.

B. Protocol for Exchanging

The decision to update any of the cryptographic components is the exclusive right of the manufacturer. We envisage three download scenarios. The first, shown in Fig.5(a) is where the software only is updated after a request from the manufacturer. In the case that the security component *also* needs to be updated, the additional security component is appended to the software which is requested, as shown in Fig.5(b). This would be used for non-urgent updating of security components. Furthermore, an urgent update can be carried out as shown in Fig.5(c). In this case no software is downloaded. At all times the user is *unaware* that the security components have been updated, that is the process is entirely automatic and transparent to the user.

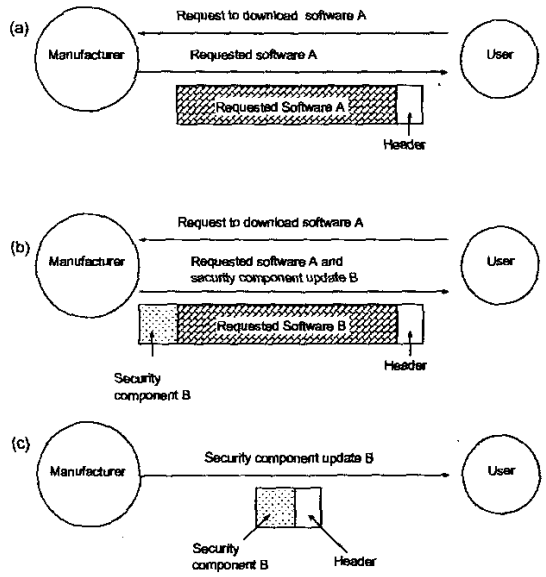

Fig. 5. Protocol diagram for downloading software and security components showing 3 cases: (a) software only download (b) software and security component download (c) security component only download.

To ensure complete security, the new cryptographic components must be encrypted during the download procedure. Since we assume that at least one component has been identified as weak, necessitating a new component download, we cannot use this component during the download procedure. That is, we cannot use the default setting of all components.

The function of the header shown in Figs.5,6 is now explained. First, in Fig.6(a) a normal software download is shown. The function of the header in this case is to show that the default cryptographic components should be used.

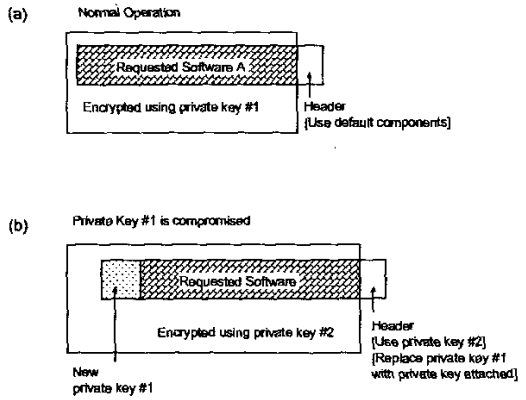

Fig. 6. Protocol diagram to replace one of the cryptographic components when it is compromised: (a) normal operation (b) replacement of compromised component

Now assume that private key #1, one of the cryptographic components is somehow compromised. The default cannot now be used for download of software or updating of the cryptographic components. Therefore, the header specifies that one of the redundant components (at least 2 components for each element must be specified at the terminal) should be used for this download. Therefore, the security of the software and update of the cryptographic component can be assured.

Later, offline the terminal can now update the compromised security component with the new one received during the software download.

V. A DISCUSSION OF CERTAIN IMPLEMENTATION ISSUES

As an illustration, the performances of certain cryptographic components are given in the following two tables.

According to [19], Table 3 shows the performance illustration on the restricted platforms like Palm Vx or Visor.

On the other hand, an illustrative comparison of the performance of certain cryptographic hash functions was carried out by computer simulation on a Pentium II 450MHz class machine running FreeBSD and processing a 766k-bytes file. The execution times are shown in Table II. The software used was C++ library routines as supplied by [20].

For specification of the employed cryptographic components see [17] or [18], for example.

Particularly, note that the results in Table II show that not all algorithms achieve the same performance, and there is a

TABLE I

PERFORMANCE OF CERTAIN CRYPTOGRAPHIC COMPONENTS ON LOW-POWER PROCESSORS [19].

cryptographic component	data	average execution time	
		Palm Vx (20MHz)	Visor (33MHz)
RSA (key: 768-bit) verify	288 bits	866ms	496ms
RSA (key: 1024-bit) verify	288 bits	1433ms	806ms
MD5	32768 bits	90ms	50ms
SHA	32768 bits	234ms	128ms
RC4	32768 bits	172ms	93ms

TABLE II

COMPARISON OF AVERAGE EXECUTION TIMES OF SOME CRYPTOGRAPHIC HASH FUNCTIONS ON A MODERATELY POWERFUL PROCESSOR ASSUMING 766K-BYTES DATA.

hash function name	av. execution time [ms]
MD5	93
SHA	231
SHA256	538
RIPMED	300

wide choice to be made as to which cryptographic primitives are employed by the manufacturer. As shown with the hash functions, a performance difference of more than five times can be seen among the different algorithms.

Also, note that the reconfigurability of cryptographic primitives may be used to provide different levels of security for different functions, all in the one device. For example, a commercial SDR terminal may become a government level secure terminal simply by changing components.

VI. CONCLUSIONS

We have discussed a scenario for the approval, confidential transmission and verification of software to be downloaded for a software defined radio system, as well as preventing usage of unapproved or illegally created software. The system is based on initial setting of the cryptographic components (by a manufacturer) which can be if necessary exchanged later in an automatic manner without any assistance or interaction from the user. Particularly note that no secure key distribution network is required. Responsibility for security of the cryptographic primitives and the keys rests mainly with the manufacturer.

Particularly, we have proposed a protocol for exchanging the employed cryptographic primitives. A discussion on implementation issues of the cryptographic components is given, as well.

REFERENCES

[1] J. Mitola, "The Software Radio Architecture", *IEEE Communications Magazine*, vol. 33, pp. 26 – 38, May 1995.

[2] M. Cummings and S. Heath, "Mode Switching and Software Download for Software Defined Radio: The SDR Forum Approach", *IEEE Communications Magazine*, vol. 37, pp. 104 – 106, August 1999.

[3] N.J. Drew and M.M. Dillinger, "Evolution Toward Reconfigurable User Equipment", *IEEE Communications Magazine*, vol. 39, pp. 158 – 164, Feb. 2001.

[4] M. Mehta, N. Drew, G. Vardoulias, N. Greco and C. Niedermeier, "Reconfigurable Terminals: An Overview of Architectural Solutions", *IEEE Communications Magazine*, vol. 39, pp. 82 – 89, August 2001.

[5] N. Nakajima, R. Kohno and S. Kubota, "Research and Developments of Software-Defined Radio Technologies in Japan", *IEEE Communications Magazine*, vol. 39, pp. 146 – 155, August 2001.

[6] *Authorization and Use of Software Defined Radio: First Report and Order*. Federal Communications Commission: Washington, D.C., Sept. 2001.

[7] L.B. Michael, M.J. Mihaljević, S. Haruyama and R. Kohno, , "A Framework for Secure Downolad for Software Defined Radio", *IEEE Communications Magazine*, vol. 40, July 2002.

[8] "Requirements for Radio Software Download for RF Reconfiguration", *Working Document SDRF-02-W-0003, Software Defined Radio Forum*, Feb. 2002.

[9] H. Shiba at. al., "Software Defined Radio Prototype for PHS and Wireless LAN Systems (I) - System Design and Over-the-air-Download -", *Technical Report of IEICE*, SR01-12(2001-10), pp. 33-38, Sept. 2001.

[10] M. Sugita, K. Uehara and S. Kubota, "Flexible Security Systems and a New Structure for Electronic Commerce on Software Radio", *52nd IEEE Vehicular Technology Conference*, Proceedings, vol. 6, pp. 3033-3040, Sept. 2000.

[11] R. Falk and N. Greco, "Securely Reconfigurable Terminals", *Proceedings, 1st Mobile Communications Summit 2001*, Barcelona, Spain, September 2001.

[12] H. Krawczyck, "The Order of Encryption and Authentication for Protecting Communications (or: How Secure is SSL)", Advance in Cryptology - CRYPTO 2001, *Lecture Notes in Computer Science*, vol. 2139, pp. 310–331, 2001.

[13] S. Fluhrer, I. Mantin and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4", Selected Areas in Cryptography - SAC2001, *Lecture Notes in Computer Science*, vol. 2259, pp. 1-24, 2001.

[14] N. Borisov, I. Goldberg and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11", 2001. http://www.isaac.cs.berkeley.edu/isac

[15] A. Biryukov, A. Shamir and D. Wagner, "Real Time Cryptanalysis of A5/1 on a PC", Fast Software Encryption - FSE2000, *Lecture Notes in Computer Science*, vol. 1978, pp. 1-18, 2001.

[16] M.J. Mihaljević and R. Kohno, "On Wireless Communications Privacy and Security Evaluation of Encryption Techniques", *IEEE Wireless Comm. and Net. Conf. - WCNC2002*, Proceedings, 865–868, March 2002.

[17] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*. CRC Press: Boca Rotton, FL, 1997.

[18] B. Schneier, *Applied Cryptography*. John Wiley and Sons Inc., 2nd Edition, 1995.

[19] V. Gupta and S. Gupta, "Experiments in Wireless Internet Security", *IEEE Wireless Comm. and Net. Conf. - WCNC2002*, Proceedings, pp. 860-864, March 2002.

[20] Wei Dai's Crypto++ Library Routines, ver. 4.2, http://www.cryptopp.com/