A Proposal for Secure Software Download in SDR

D.S.Dawoud
University of KwaZulu Natal
Email dawoudd@nu.ac.za

Abstract:
To promote the commercial implementation of software defined radio (SDR) terminals, a secure method of download is vital. The flexibility of the system to change from one environment to the other and the possibility of changing any cryptographic component at any time is also an important issue In this paper we are discussing three main design metrics of SDR. A proposed protocol for changing any cryptographic component is proposed.

1. Introduction

A Software Defined Radio (SDR) terminal may be regarded as a programmable radio transceiver whereby the user equipment is able to reconfigure itself, in terms of its capability, functionality and behaviour in order to dynamically accommodate the needs of the user. It is expected that SDR as a technology will help bring together the different forms of communications. The incorporation of mobile communications, broadcast receivers, location services, internet, multimedia, dedicated point-to-point communications, personal computing, and digital aids (PDAs) would all be possible with the help of a mature and reliable SDR technology. This would eventually lead to the realization of reconfigurable radio systems and networks that would consist of self-organizing, self-evolutionary intelligent radio [2] system infrastructure and user terminals for *ubiquitous information interaction.*

In its mature form, the SDR terminal [1] will be able to reconfigure itself at any level of the radio protocol stack by implementing appropriate software within an adaptive hardware platform. Downloading appropriate software, a SDR terminal will be able to:

- *Adapt its behaviour* -- To change the applications, range, services and functionality of the terminal to meet the demands of the user in accordance with the capability of the terminal.
- *Traverse across different communication standards* -- This will allow the terminal to switch between *different modes. This would involve reconfiguration* of the radio protocol stack, change in data rates, and different RF band and carriers.
- *Evolve with user demand* – In order to provide the user with desired services, at the time of need, at a cost they can afford, while maximizing the quality of service (QoS) delivered.

Successful downloading is measured by many metrics. This paper is considering three of them: security, flexibility and performance.

In section-2 we are giving a background on the security in mobile system. In section-3 we are discussing the performance metric and in section-4 we discuss the flexibility metric. In section-5 we are proposing a protocol for updating of cryptographic components. The main conclusions are given in section-6.

2. Security Concerns in Mobile Systems

The role of security mechanisms is to ensure the privacy and integrity of data, and the authenticity of parties involved in a transaction. In addition, it is also desirable to provide functionality such as non-repudiation, copy protection, preventing denial-of-service attacks, filtering of viruses and malicious code, and in some cases, anonymous communication [2,3].

Some of the major security concerns from the perspective of a mobile appliance are:

- *User identification* attempts to ensure that only authorized entities can use the appliance.
- *Secure storage* addresses the security of sensitive information such as passwords, PINs, keys, certificates, *etc.*, that may reside in secondary storage (*e. g.*, flash memory) of the mobile appliance.
- A *secure software execution environment* is necessary to ensure that attacks from malicious software such as viruses or trojan horses are prevented.
- A *tamper-resistant system implementation* is required to ensure security of the hardware implementation from various physical and electrical attacks.
- *Secure network access* ensures that only authorized devices can connect to a network or service.
- *Secure data communications* considers the privacy and integrity of data communicated to/from the mobile appliance.
- *Content security* refers to the problem of ensuring that any content that is downloaded or stored in the appliance is used in accordance with the terms set forth by the content provider (*e. g.*, read only, no copying, *etc.*).

Wireless data communications can be secured by employing security protocols that are added to various layers of the network protocol stack, or within the application itself. Security protocols utilize cryptographic algorithms (asymmetric or public-key ciphers, symmetric or private-key ciphers, hashing functions, *etc.*) as building blocks in a suitable manner to achieve the desired objectives (peer authentication, privacy, data integrity, *etc.*).

Many of these protocols address only network access domain security, *i.e.*, securing the link between a wireless client and the access point, base station, or gateway.

0-7083-8605-1 / $17.00 2004 IEEE

Several studies have shown that the level of security provided by most of the above security protocols is insufficient, and that they can be easily broken or compromised by serious hackers [4, 5].

While some of these drawbacks are being addressed in newer wireless standards such as 3GPP [6] and 802.11 enhancements [7], it is generally accepted that:

- The wireless standards need to be complemented through the use of security mechanisms at higher protocol layers.
- The security measures must be distributed between the different players involved in the system (handset, base station, vendor, etc.) and it is imperative to take a hierarchical approach where each layer of security provides a foundation for the one above it.
- Practically speaking no one believes that there is any solution to SDR security which doesn't involve the application of software as part of the threat mitigation strategy. So software is necessary but is it sufficient? We assert that SDR Security with any degree of confidence will require some elements to be enforced by hardware measures.
- Combining hardware and software crypto components plays a significant role in providing a strong crypto foundation that meets the basic security requirements mentioned above (i.e. authentication, confidentiality, integrity, and non-repudiation).

To make the assertion more directly, not only *can* security mechanisms be implemented in a hardware module, they *must be* to prevent tampering. The envisioned hardware mechanisms include a processing core, protected internal memory, and additional features necessary to implement whatever security measures are standardized

2.1 Tamper-Resistance

Most security protocols and mechanisms address security of a mobile appliance without regard to the specifics of the implementation. Theoretical analyses of the strength of cryptographic algorithms assume that malicious entities do not have access to the implementation (classical cryptanalysis). Here, a cryptographic primitive is viewed as an abstract mathematical object, that is, a mapping of some inputs into some outputs parameterized by a secret value, called the key. An alternative view of the cryptographic primitive comes from its implementation. Here, the primitive manifests itself as hardware circuit or as a program that will run on a given embedded processor, and will thus present very specific characteristics. Such a view implies that security protocols and cryptographic algorithms can simply be broken by observing properties of the implementation (for example, "side-channel information", such as timing, power, magnetic field, behavior in the presence of faults, *etc.*), by "cloning" (one-to-one copy) of a cryptographic algorithm together with its key, or by Readback attack (a feature provided for most FPGA families). In some cases it can be enough to run the cloned application in decryption mode to decipher past and future communications. Sensitive data can also be compromised, while it is being communicated between various components of the system through the on-chip communication architecture, or, even when simply stored in the mobile appliance (in secondary storage like Flash memory, main memory, cache, or even CPU registers).

Thus, secure design of the HW/SW system architecture becomes as important as secure data communications. We will now give a brief overview of the common techniques that can be used to "attack" a mobile device. The discussion assumes mostly that an attacker has physical access to the encrypted device. The techniques are classified into two broad categories: *physical and side-channel attacks, and, software attacks.*

Physical and side-channel attacks refer to attacks that exploit the system implementation and/or identifying properties of the implementation. It is not surprising that the first target of these attacks [8, 9] is mobile devices such as smart cards. For concreteness, the discussion here will be put in that context, although most of it applies to other (cryptographic) devices as well. Physical and side-channel attacks are generally classified into invasive and non-invasive attacks. Invasive attacks such as micro-probing techniques involve getting access to the silicon to observe, manipulate and interfere with the system internals. The aim of a physical attack is to investigate the chip design to get information about proprietary algorithms or to determine the secret keys by probing points inside the chip. In case of FPGAs, these attack targets parts of the FPGA, which are not available through the normal I/O pins. This can potentially achieved through visual inspections and by using tools such as optical microscopes and mechanical probes. Since invasive attacks typically require relatively expensive infra-structure, they are much harder to deploy and will only be possible for large organizations, for example intelligence services. Non-invasive attacks, on the other hand, do not require the device to be opened. While these attacks require knowledge of the system, they tend to be cheap and scalable (compared to invasive attacks).

There are many forms of non-invasive attacks. Any physical implementation of a cryptographic system might provide a side channel that leaks unwanted information. Examples for side channels include in particular: power consumption, timing behavior, and electromagnetic radiation. Fault induction techniques manipulate the environmental conditions of the system (voltage, clock, temperature, radiation, light, eddy current, *etc.*) to generate faults and to observe the related behavior [10]. Eavesdropping techniques attempt to deduce information by monitoring any accessible system resources such as the supply and interface connections. The most common form of this attack involves analyzing the power consumption of the system [11]. In such cases the attacker may use Simple Power Analysis (SPA) and Differential Power Analysis (DPA) to analyze the power consumption of the device while performing a cryptographic operation in order to find the secret keys from a tamper resistant device. The main idea of DPA is to detect regions in the power consumption of a device which are correlated with the secret key. Other possibilities involve analyzing the electromagnetic radiation around the device [12]. Another important class of attacks is the timing attack [12], which exploits the observation that the computations performed in some of the cryptographic algorithms often take different amounts of time on different inputs. A well-known example is the implementation of the RSA public-key cryptosystem using the Chinese Remainder Theorem (CRT) for improving the performance. Other attacks targeting symmetric encryption schemes such as DES have also been used.

In recent years, there has been a lot of work done to prevent side-channel attacks. The proposed methods can generally be divided into software and hardware countermeasures, with the majority of proposals dealing with software countermeasures. "Software" countermeasures refer primarily to algorithmic changes, such as masking of secret keys with random values, which are also applicable to implementations in custom hardware of FPGA. Hardware countermeasures often deal either with some form of power trace smoothing or with transistor-level changes of the logic.

Software attacks are based on malicious software being run on the mobile terminal, that exploits weaknesses in security schemes and the system implementation. The likelihood of software attacks tends to be high in systems such as mobile terminals, where application software is frequently down-loaded on-the-air. The downloaded software may originate from a non-trusted source and, hence, can be used to implement attacks. Compared to physical attacks, software attacks typically require infrastructure that is substantially cheaper and easily available to most hackers, making them a serious immediate challenge to secure system design.
Building attack resistance especially into software [13] would necessitate one or more of the following measures: (i) finding a means to ascertain the operational correctness of protected code and data, before and during run-time, (ii) providing protection against trojan horse applications trying to steal data (*e.g.*, cryptographic keys) from a security application that is run on behalf of the user, (iii) enforcing that application content can remain secret (*digital rights management*), and (iv) protecting against probing (looking at the memory used by secure applications) and reverse engineering (de-compilation, flow analysis, profiling *etc.*).

3. Computational Requirements of Security Processing: *Performance Gap*

Current protocols use a number of cryptographic algorithms, including symmetric ciphers, one way hash functions, and asymmetric ciphers. An asymmetric cipher — public key encryption — depends on some very difficult computational problem (like finding the prime factors of extremely large numbers as in the RSA algorithm) to ensure that only the intended recipients can decode encrypted data, thus decoding is expensive even when the appropriate information is known. Symmetric ciphers are based on the involved parties having private information that allows data to be decoded. The difficulty with symmetric ciphers is getting the private information to the parties uncompromised— the key distribution problem. However, once the private key is distributed, the necessary computation is manageable. In many systems, asymmetric ciphers (such as RSA, DSA, ECC, and Diffie–Hellman) are used to encrypt keys for use in symmetric ciphers such as AES, DES, IDEA, CAST, etc. Hash functions can be used to verify data integrity. However, even when using symmetric cipher algorithms to encrypt data, it is still quite difficult (or expensive) to implement an embedded system based on a general–purpose processing unit that can process encrypted data at commonly used rates. For example, a copper–based LAN can operate at rates from 10 Mbps to 10 Gbps (and will soon be able to operate at 40 Gbps!) and wireless LANs operate in the 2-54 Mbps range. The latest embedded processors would only barely be able to keep pace at about 4 Mbps with full utilization. This is not useful if the system processes encrypted video, high–bandwidth multimedia, or an otherwise rate–demanding application.
Thus, there exists a clear mismatch between the security processing requirements and the available processor capabilities, even if the workload of the appliance is assumed to be completely dominated by security processing. In other words, this mismatch is likely to be worse in reality since the processor is typically burdened by a workload that also includes other application software, network protocol and operating system execution.
This mismatch leads to a *"wireless security processing gap"*. While embedded processor performance can be expected to increase due to improvements in fabrication technologies and innovations in processor architecture, the increase in data rates (due to advances in wireless communication technologies), and the use of stronger cryptographic algorithms (to stay beyond the extending reach of malicious entities) threaten to further widen the wireless security processing gap.

3.1 Battery Gap

The computational requirements of security protocols stemming from the inherent complexity of cryptographic algorithms suggest that the energy consumption of these algorithms will be high. For battery powered mobile appliances, the energy drawn from the battery directly impacts the system's battery life, and, consequently, the duration and extent of its mobility and its overall utility. To illustrate the impact of security processing on battery life, the author of [14] showed that the energy cost of transmitting an encrypted message was about 200% more than the cost of transmitting an unprotected signal in a wireless sensor network.
One solution to the problems of performance gap and battery gap, is to use more and more powerful CPU and let software to handle the encryption.
However, this solution is very costly. Another solution that current industry is leaning towards is to have a special coprocessor and let it handle the encryption work. This is a potentially cost effective solution and there have been several approaches. We will discuss these approaches as well as their shortcomes in the next section.

4. Flexibility

A fundamental requirement of a mobile appliance is the ability to cater to a wide variety of security protocol standards in order to facilitate interoperability in different environments. For example, an appliance that needs to work in both 3G cellular and wireless LAN environments would need to execute security algorithms specified by 3GPP [6] as well as the Wired Equivalent Privacy (WEP) algorithm specified by the 802.11 standard [7]. Additionally, a device is often required to support distinct security processing standards at different layers of the network protocol stack. For example, a wireless LAN enabled PDA that supports secure web browsing may need to execute both WEP (Link Layer) and SSL (Transport Layer), while the same PDA, if required to connect to a virtual private network

(VPN), may additionally need to support IPSec (Network Layer).

The implementation of such flexibility needs the use of security protocols which typically allow for the usage of a wide range of cryptographic algorithms. Normally to ensure secure downloading the protocol employs three cryptographic primitives, a hash function, a digital signature and symmetric key ciphering, as well as secret key and a public key. Many protocols are available in the literatures, which support the use of different ciphers for its operations (authenticating the server and client, transmitting certificates, establishing session keys, *etc.*). SSL [15], IPSec, MET are some examples. SSL, for example allows the use of cryptographic algorithms such as RSA and KEA as possible choice for key exchange. For symmetric encryption, an RSA key exchange based SSL cipher suite would need to support 3-DES, RC4, RC2 or DES, along with the appropriate message authentication algorithm (SHA-1 or MD5). Since the mobile appliance may have to communicate with a server/client that uses a specific combination of cipher suite and key exchange algorithm, it is desirable to support all the allowed combinations so as to inter-operate with the widest possible range of peers.

Another very important reason for designing flexible security protocols is the tendency toward revising the protocol standards to enable new security services, add new cryptographic algorithms or drop weaker ciphers. The possibility of changing any of the cryptographic components employed (i.e. drop weaker cipher and replacing it with stronger one) is motivated by the reality that the security evaluation of currently available cryptographic techniques can point out the weaknesses in a technique, but can not yield a definitive proof of security. In other words, if 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. An illustrative example is the A5 cipher used in GSM. After a number of years and widespread use of A5, serious weaknesses and its break-ability have been reported.

Security evaluation of the cryptographic primitives is recognized as a very important issue, and it is the main topic of a number of international projects including NESSIE and CRYP-TREC. Results of these and related projects [8] are a strong indication of the need to include the possibility to change, modify or update some or all of the cryptographic components.

Concerning designing flexible protocols, we can refer here to the continuous modifications happening to the well-established protocols as TLS, WTLS, and MET, which aim to give the required flexibility to the protocols. However, it is anticipated that future security protocols would be specifically tailored from scratch for the wireless environment. This presents a formidable challenge to the design of a security processing architecture, since flexibility and ease-of-adaptation to new standards become equally important design considerations as traditional objectives such as power, performance, *etc.*

5. Exchanging of Cryptographic Components

The proposed system for secure downloading employs three cryptographic primitives, a hash function, a digital signature (can be more than one signature, e.g. manufacturer signature, vendor signature, etc.) and symmetric key ciphering, as well as a secret key and a public key during each downloading procedure. Our proposal includes the possibility to change any of the cryptographic components employed. As mentioned before, inclusion of this possibility was motivated by:

1. The need for flexibility to facilitate interoperability in different environments.
2. The reality that the security evaluation of currently available cryptographic techniques can point out the weaknesses in a technique, but can not yield a definitive proof of security. As a result of this fact, if the manufacturer or the vendors, at a later stage, identify weaknesses in any security component or that one of them is broken by some cryptanalysis techniques they have to change them automatically without informing the user.
3. Changing the security keys from time to time increases the security of the system and gives the possibility of neutralizing some possible attackers.
4. The possibility of designing and implementing an Application Specific Encryption Processor (ASEP) that incorporates multiple algorithms and sessions to achieve algorithm concurrency. ASEPs are normally tamper-resistance especially if it is implemented as application specific integrated circuit and not on the basis of FPGA.

An illustration of the collection of cryptographic components used in the proposed system is displayed in Table 1.

Therefore, in our proposal we include a method for exchanging any of the cryptographic primitives or keys with a new one in an automatic manner, which does not require the interaction of the user.

5.1 Model of Exchanging

We propose a matrix model of exchangeable cryptographic components. The model is based on the following:

1. The proposed system for secure downloading employs three cryptographic primitives, a hash function, a digital signature and symmetric key ciphering, as well as a secret key and a public key during each downloading procedure.
2. Instead of looking to the key as a single entity that is represented by one bitstream of n bits, we are proposing to split each key into k sub-streams each of m bits ($k = n/m$) and we look to each substream as separate security entity.
3. The manufacturer selects and stores a number of substreams to be stored in the form of an $q \times k$ array. To form a key, we select one element from each column. The selected elements form a path of k nodes. By this way it is possible to have a total of qk possible keys stored in the device. The manufacturer, through downloading, can switch from one key (one path) to another one. He can also change completely one of the stored keys by changing the sub streams that form the corresponding path.
4. Each key substream is considered as separate security entity (separate cryptographic component).
5. In Table-1, all cryptographic components employed are considered as elements of an $q \times (3 + 2k)$ matrix where each column corresponds to a collection of $q > 2$ elements for each of the cryptographic components employed: hashing functions, digital signatures, ciphering algorithms, secret keys and public keys.

0-7083-8605-1 / $17.00 2004 IEEE

6. The first row of the matrix contains those cryptographic components which are used by default.
7. Exchanging a cryptographic component is equivalent to updating a matrix entry.
8. Selection of a combination of the cryptographic components for updating is a selection of a path in the matrix. Accordingly, more rows (i.e. increasing the value of q) in the matrix yields the opportunity for more choices for selection of a path which yields the desired level of security.
An illustration of the cryptographic components collections is given in Table-1.

5.2 Proposal for a Model of Exchange

The goals of the proposal for the changeability of the cryptographic components include:

- exchange should be an automatic procedure;
- it should not require any assistance of the user, and more particularly
- the user should not be aware that an exchange has been performed.

The underlying ideas for exchanging cryptographic components include the following.

- It is assumed that in each time instant at least one element from each of the collections of cryptographic components can be considered secure.
- The software downloading protocol should support downloading of alternative cryptographic components.
- The procedure for updating a certain element of each of the collections is usually carried out at the same time as program software is downloaded.

5.3 Protocol for Exchanging

The downloading protocol includes information whether the default cryptographic primitives and keys should be employed or other selections should be made. This is specified in a header for the downloaded software. The general format of the protocol is shown in Fig.1. The protocol consists of three fields: header to specify if the system is going to use the default cryptographic primitives or there are changes, the requested software (if any), and the third field contains information about the new path (if any) of the cryptographic entities (or the updated entities) that will replace the default path.

For example encryption primitive #1 is used by default, but a weakness is discovered. For the next program file software download, the header will inform the system that there are changes in primitive #1 and the last field of the protocol will contain description for the new path that contains primitive #2 instead of #1. In this case the system will use the new path for decrypting the downloaded software and considers path #2 as default. In case if primitive #2 is not part of our array and it represents an updating of an element in the array, in this case the manufacturer can use urgent update mode to replace #1 by #2.

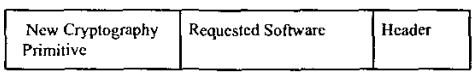

Fig. 1 Protocol Format

The decision to update any of the cryptographic components is the exclusive right of the manufacturer. The updating can be:

1. Non-urgent updating: The updating occurs during the next request from a terminal for downloading of any software. In this case the additional security component is appended to the software which is requested. The terminal request can be either asking for new application or while moving from one environment to another.

2. Urgent updating: The manufacturer identifies the occurrence of break down of one or more cryptographic primitives and he has to respond quickly by downloading the new cryptographic primitives. In this case no additional software is downloaded.

The above two cases are shown in Fig.... 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. Beside the above two download scenarios, there is the normal scenario, i.e. the case where the software is updated either by the manufacturer or by the terminal when requesting new application. To discriminate between the different scenarios and to define exactly which security component is going to change, a header is included with the software.

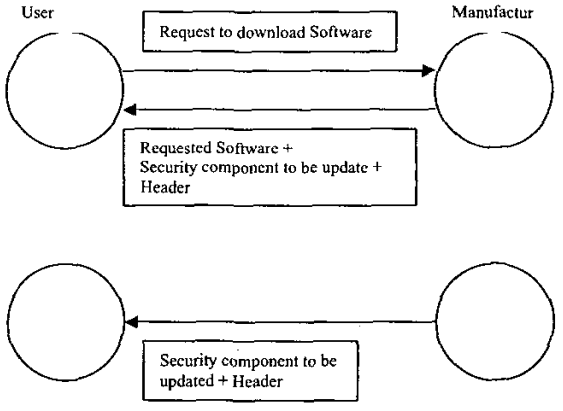

Fig. 2 Protocol diagram

The basic procedure for updating is as follows.

1. *Phase I: Offline steps at the manufacturer*
a. the manufacturer decides to update a certain primitive, due to some weakness that is discovered
b. the manufacturer selects a different combination of cryptographic components from those already available on the software radio terminal to be used for the next software down-load and update of the compromised cryptographic primitive.
2. *Phase II: Online phase involving manufacturer and SDR terminal*

a. upon receiving the request from a SDR terminal for downloading of program software, the manufacturer downloads the requested program software using the new security components selected in step I(b). The information as to which components are used would usually be contained in the download protocol header.

b. in addition to the requested software the manufacturer additionally and unknown to the user also downloads any software for updating the weak cryptographic components

3. *Phase III: Off-line processing at the terminal side*

After the downloading the new cryptographic primitives, the updating is performed in an off-line manner.

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.

The importance of our proposal of storing more than one key can be explained here. Assume that the manufacturer identified that the default key 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 available keys 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 previous software download.

6. Conclusions

In this paper we discussed three design metrics that measures the success of software downloading. We proposed a protocol to increase the flexibility of the system and give the possibility of changing any cryptographic element automatically.

7. References

[1] *MeT PTD definition (version 1.1)*. Mobile Electronic Transactions Ltd. (`http://www.mobiletransaction.org/`), Feb. 2001.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 1998.

[3] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley and Sons, 1996.

[4] Y. Frankel, A. Herzberg, P. A. Karger, H. Krawczyk, C. A. Kunzinger, and M. Yung, "Security issues in a CDPD wireless network," *IEEE Personal Communications*, vol. 2, pp. 16–27, August 1995.

Proceedings of the Design,Automation and Test in Europe Conference and Exhibition (DATE'03)
1530-1591/03 $17.00 © 2003 IEEE

[5] S. Patel, "Weaknesses of North American wireless authentication protocol," *IEEE Personal Communications*, vol. 4, pp. 40–44, june 1997.

[7] *3GPP Draft Technical Specification 33.102, 3G Security Architecture*. `http://www.3gpp.org`.

[8] R. Anderson and M. Kuhn, "Tamper resistance - a cautionary note," 1996.

[9] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *IWSP: International Workshop on Security Protocols, LNCS*, 1997.

[10] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults," *Springer-Verlag Lecture Notes in Computer Science (Proceedings of Eurocrypt'97)*, vol. 1233, pp. 37–51, 1997.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Springer-Verlag Lecture Notes in Computer Science*, vol. 1666, pp. 388–397, 1999.

[12] W. van Eck, "Electromagnetic radiation from video display units: an eavesdropping risk?," *Computers and Security*, vol. 4, no. 4, pp. 269–286, 1985.

[13] D. Aucsmith, "Tamper Resistant Software: An Implementation," *Information Hiding, Springer Lecture Notes in Computer Science*, vol. 1174, pp. 317–333, 1986.

[14] D. W. Carman, P. S. Krus, and B. J. Matt, "Constraints and approaches for distributed sensor network security," Tech. Rep. #00-010, NAI Labs, Network Associates, Inc., Glenwood, MD, Sept. 2000.

[15] *SSL 3.0 Specification*. `http://wp.netscape.com/eng/ssl3/`.

Table-1 (3 + *2k*) collections of the cryptographic components. (The keys can be split into sub-primitives)

Entry	Hashing Primitive	Digital Signature Primitive	Encryption Primitive	Secret Key (k substreams)	Public key (k substreams)
# 1	SHA-1	RSA based	DES (any))	Secret key 1	Government
#2	MD5	ECC based	IDEA (any)	Secret key 2	Manufacturer
#3	SHA-512	DL- based	AES (any)	Secret key 3	Manufacturer
#4	Proprietary algorithm X	Proprietary algorithm Y	Proprietary algorithm Z	Secret key 4	Public key Y