S

Salt

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Dictionary Attack; ►Dictionary Attack (I); ►One-Way Function; ►Password

Definition

A *salt* is a random string used in conjunction with a password to make offline password guessing attacks more difficult.

Applications

A *salt* is a t-bit random string that may be prepended or appended to a user's ►password prior to application of a ►one-way function in order to make ►dictionary attacks less effective. Both the salt and the hash (or encryption) of the augmented password are stored in the password file on the system. When the user subsequently enters a password, the system looks up the salt associated with that user, augments the password with the salt, applies the one-way function to the augmented password, and compares the result with the stored value [1–3].

It is important to note that the work factor for finding a particular user's password is unchanged by salting because the salt is stored in cleartext in the password file. However, it can substantially increase the work factor for generating random passwords and comparing them with the entire password file, since each possible password could be augmented with any possible salt. The effort required to find the password associated with an entry in the password file is multiplied by x (where x is the smaller of {the number of passwords; 2^t}), compared with a password file containing hashes (or encryptions) of unsalted passwords [3].

Another benefit of salting is that two users who choose the same password will have different entries in the system password file; therefore, simply reading the file will not reveal that the passwords are the same.

Recommended Reading

1. Denning D (1982) Cryptography and data security. Reading, MA, Addison-Wesley
2. Kaufman C, Perlman R, Speciner M (1995) Network security: private communication in a public world. Englewood Cliffs, NJ, Prentice-Hall
3. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. Boca Raton, FL, CRC

Sandbox

Mark Stephens
SMU HACNet Labs, School of Engineering, Southern Methodist University, Nacogdoches, TX, USA

Related Concepts

►Virtualization; ►Web Browser Security and Privacy; ►Web Service Security

Definition

Sandboxing is a technique for enforcing security policies on untrusted guest applications in a secure environment (i.e., "sandbox") to eliminate risk to a host system.

Background

In general, the term "sandbox" or "sandboxing," when applied to Computer Science or Information Technology, can describe any method for creating restricted environments (i.e., a sandbox) for untrusted guests. For example, Romney and Stevenson [1] describe the deployment of a sandbox network laboratory, which is designed, maintained, and operated by students studying security systems engineering.

Although the concept of sandboxing can be applied to any situations where one is attempting to provide a safe and restricted environment, in recent years the term is most commonly associated with methods for providing access to untrusted software, albeit restricted, on host systems for execution. As such, the rest of this entry will assume the use of the term "sandbox" to specifically mean securing untrusted application execution on host systems.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,

The earliest application sandboxing results were explored by the fault tolerance research community, e.g., software fault isolation (SFI). The concept of SFI sandboxing for fault tolerance purposes was first introduced by Wahbe et al. [2]. Thus the earliest sandbox research was focused on the identification and isolation of software bugs rather than malicious software (e.g., virus, worms, etc.).

In the 1990s, application sandboxing was identified as an imperative solution for solving fundamental security problems that arise in distributed computing. In short order, distributed computing security became the primary impetus for application sandboxing research, and thus the dominant contributions in academic literature of the same. No doubt this is in large part due to the proliferation of distributed computing technology over the past two decades, e.g., Web Computing, Distributed Operating Systems, Grid Computing, Mobile Agents, Peer-to-Peer (P2P) Computing, etc. Rubin and Geer [3] provide a nice high-level introduction to security challenges and techniques used when attempting to secure mobile agent solutions in a distributed computing environment.

Common approaches to sandboxing untrusted guest applications include *Operating System Methods*, *Virtualization*, *Process Wrappers*, and *Binary Re-writing*. A typical sandbox solution will employ one or more of these methods to ensure untrusted guest applications are well behaved when executing on host systems. Any security policy violation on behalf of the untrusted application typically results in immediate application termination. Each of these approaches will now be described in more detail.

The use of *Operating System Methods* for sandboxing leverages the OS security mechanisms to ensure untrusted guest applications do not violate host security policies. This can include leveraging user level security policies and/or execution prioritization.

The use of *Virtualization* for sandboxing is handled via *Virtual Machines*. *Virtual Machines* can be applied at the operating system or process layer. Operating system layer virtualization allows for different "Virtual Machines" to co-exist on the same hardware and share system resources. Each virtual machine can execute different versions of the same operating system or entirely different operating systems all together. For example, a single server may have both a Linux and a Windows virtual machine executing in parallel, sharing system-level resources (e.g., hard drives, CPU, RAM, etc.) while limiting access to virtual resources (e.g., files, directories, etc.). Examples of commercial system-level virtual machine vendors include VMWare, Citrix/XenSource, Microsoft, and Novell.

Process Virtual Machines inject a VM interpreter process between the OS and the application layers. Thus, when an application is targeted for execution, the OS actually launches the VM process, which in turn launches and monitors the application. This approach of virtualization was popularized with the advent of the Java Programming Language. A Java program is compiled to a non-OS target instruction set called *Java bytecode*. The *Java bytecode* is then in turn executed by the *Java Virtual Machine* (*JVM*). JVMs are written and compiled to the specific target OS. Microsoft's competing solution to Java bytecode and the JVM are *Common Intermediate Language (CIL)* and *Common Language Runtime (CLR)*, respectively.

Arguably, the use of *Process Wrappers* for sandboxing is a special case of process-level virtualization. The primary difference between the two is that process wrappers are custom-built sandbox VMs that focus specifically on containment of untrusted application execution on host systems, where commercial Process Virtual Machines provide additional benefits to include OS independence, Just-In-Time (JIT) compilation, memory management and garbage collection, etc.

Similar to SFI, *Binary Re-writing* finds its research roots in code instrumentation for purposes of optimization, fault tolerance, and debugging. Only after the need for sandboxing as an enabler for Distributed Computing technology was identified did these techniques find new purpose in security policy enforcement of the same. Binary re-writing for sandbox purposes involves the injection of new and/or replacements of existing binary instructions in the compiled code such that a host security policy can be enforced. For example, before an application is allowed to be launched by the OS the executable is sent through a binary re-writing utility that identifies and replaces system-level calls in order to restrict access to specific host resources. Related topics include DLL-injection, rootkits, API Hooking, Kernel Hooking, and Runtime Patching. A definitive how-to-guide on subverting security in the Windows OS is the text by Hoglund and Butler [4]. This text is specifically very useful for any student or practitioner who is interested in developing sandboxing technology for the Windows OS.

Theory

Process layer virtualization is the most common form of VM sandboxing. The most famous example of process layer virtualization being the Java Applet, which leverages the JVM to place mobile java bytecode downloaded from the Web (i.e., Applets) in a virtual sandbox. This virtual sandbox enforces the security policy on the untrusted

code, e.g., restricting access to file system, not allowing outside network communication except back to the Web server that provided the page containing the Applet, etc.

Proposals for system-level virtualization have been less prevalent than that of process virtualization. To leverage system-level virtualization as a sandbox technique, the host systems cannot be general purpose user computers; rather the host systems must be part of a specifically dedicated environment for spinning up and down VMs on demand. Santhanam et al. [5] proposed this model with an excellent description of the benefits, challenges, and overhead of the same.

The more specialized form of process-layer virtualization known as process wrappers has gained widespread adoption due to the ability to custom tailor the virtualization toward specific needs with little runtime overhead. Ford and Cox [6] proposed a lightweight user-level sandboxing approach specific for the x86 architecture. The Ford and Cox approach leverages x86 segmentation hardware to sandbox the untrusted application's data access while using lightweight instruction translation to sandbox the application's instructions. The untrusted application may be written in any language since the sandbox executes straight x86 instructions. The result is a very lightweight and portable multi-OS solution that shows widespread application since the x86 is currently the predominant processing architecture for desktop, laptop, and server computing. The Ford and Cox approach exploits hardware support available in the 32-bit family of the x86 architecture, thus in theory being applicable to any modern 32-bit OS version for the x86 (e.g., Linux, FreeBSD, Mac OS X, Windows, etc.). However, since their method for sandboxing leverages protected-mode segmentation, which has been disabled with the recent 64-bit extension of the x86, the Ford and Cox approach cannot be leveraged to execute 64-bit applications; however, it can still be used to execute 32-bit applications executing on a 64-bit architecture.

Many sandbox models leverage built-in security mechanisms at the OS layer combined with process wrappers to restrict unauthorized access to host resources by untrusted applications. One of the earliest papers to describe this model was proposed by Jensen and Hagimount [7]. Jensen and Hagimount describe the deployment of application "wrappers" to execute untrusted code as a non-privileged user. This simple technique relies on the fact that most multi-user operating systems determine an executable's access rights to system resources via applying the security policy of the user that launches the said executable. Thus, to restrict access to untrusted applications, Jensen and Hagimount's "wrapper" launches the executable as a non-privileged user with little or no access to system resources. Thus this approach is entirely dependent on the OS to enforce the sandbox, while incurring very little overhead. However, this approach does have significant limitations. First, the approach as described specifically restricts implementation on Windows or Mac OS. Second, it requires custom "wrapper" sandboxes to be compiled for each untrusted guest application seeking execution on the host system. And third, many advances in Rootkit research have identified methods by which malicious applications can find ways to "promote" their execution security policy by both direct and indirect methods, thus allowing access to otherwise restricted system resources.

The earliest sandbox paper leveraged the binary rewriting model for the purpose of Software Fault Isolation (SFI). This paper was presented by Wahbe et al. [2]. Their approach required the code and data to be loaded into a separate portion of the application's address space, which was specifically partitioned for untrusted code, i.e., the "sandbox." The untrusted object code is then scanned and modified to enforce specific security policies to include prohibiting jumping or writing to an address space outside of the sandbox. This approach allows for faster intermodule communication at the cost of longer execution times for untrusted modules. This approach was later expanded to be more applicable for security applications to include rigorous malware checks on the CISC architectures by McCamant and Morrisett [8].

Song and Fleisch [9] proposed a unique binary rewriting approach that integrates Java bytecode re-writing capabilities with Web caching proxies. The intention is not to replace the intrinsic security capabilities provided by the JVM, rather expand upon the same. This result allows system administrators to tailor and custom security policies beyond those capabilities provided by the JVM and/or provide workarounds for known JVM security issues. The obvious limitation of this sandboxing approach is that it only is applicable for those untrusted guest applications written in Java.

Applications

Of primary interest in distributed and grid computing is the ability to leverage multi-domain distributed computational resources as a single ubiquitous resource for solving a common problem. This includes harnessing idle CPU cycles, accessing special purpose hardware, and leveraging distributed storage. This ambitious goal has many challenges, not the least of which is how to allow untrusted software to leverage shared computational resources in a

safe and secure manner. To this end, most distributed and grid platforms leverage sandboxing in some form.

Frey et al. [10] proposed a method they dubbed mobile sandboxing to address sharing of resources in a multi-site environment using the famous Condor and Globus Grid Computing platforms (i.e., Condor-G). Mobile sandboxing enables end-users to formulate requests for very specific execution environments on remote resources. The mobile sandboxing technique addresses issues that arise with system and site policy heterogeneities, e.g., local security policies that prohibit access to file systems and/or enforce restrictions on execution times of remote jobs. Mobile sandboxing is implemented by having a generic grid daemon (i.e., wrapper) executing on the local resources, which advertise to the grid the local node's capabilities and availability. The daemon is responsible for downloading, starting, and monitoring individual tasks from the grid that require resources. The daemon uses system call trapping that redirects system calls issued by a task back to the originating system. In this manner, idle CPU cycles can be harnessed on remote nodes for remote task execution, even when each node has different local security policies. This also allows for checkpointing of work so that tasks can be migrated to other systems once a task has reached its limit for allowed execution time.

The experimental data that is collected and processed from the Large Hadron Collider at CERN is immense. The LHC storage and computational requirements necessitate the application of distributed and grid computing technology across many collaborative organizations around the world. As such, this critical research in experimental physics demonstrates the importance of sandbox model applications to solve real world problems. The paper by Andreeva et al. [11] provides a very interesting case study of this application of grid computing to include their implementation of sandbox models.

Open Problems

A silver bullet sandbox solution would meet the following criteria:

1. Allow system owners to define individual security policies for executing untrusted applications on their host systems
2. Enforce the said security policies without failure
3. Support all modern OS without restriction or kernel modifications
4. Enforce security policies to all applications in a completely unobtrusive manner

Although these criteria are easy to understand, they are almost intractable to implement 100%. The current directions in developing new sandbox techniques rely primarily on virtualization to abstract away OS constraints and binary re-writing to minimize overly restrictive programming requirements on behalf of the untrusted application developers, which most likely will not be aware of the distributed environment their software will be executing. As with many areas of interest in computer science – there appears to be no silver bullet. Rather the researcher or practitioner who is required to implement sandboxing must consider the specific needs of their solution and consider the models discussed herein from virtualization to binary re-writing, weighing the cost and benefits of each, realizing that a hybrid approach is what is most commonly employed.

Recommended Reading

1. Romney GW, Stevenson BR (2004) An isolated, multi-platform network sandbox for teaching IT security system engineers. Proceedings of the 5th conference on information technology education, SIGITE '04 ACM, Salt Lake City, UT, USA, October 28–30, 2004
2. Wahbe R, Lucco S, Anderson TE, Graham SL (1993) Efficient software-based fault isolation. Proceedings of the 14th symposium on operating systems principals, SIGOPS '93 ACM, Asheville, North Carloina, December 5–8, 1993, pp 203–216
3. Rubin AD, Geer DE Jr (1998) Mobile code security. IEEE Internet Computing 2:30–34
4. Hoglund G, Butler J (2006) Rootkits – subverting the windows kernel. Addison-Wesley, Pearson Education, Inc, MA
5. Santhanam S, Elango P, Arpaci Dusseau A, Livny M (2005) Deploying virtual machines as sandboxes for the grid. In: Second workshop on real, large distributed systems. WORLDS '05 USENIX, San Francisco, CA, December, 2004
6. Ford B, Cox R (2008) Vx32: lightweight user-level sandboxing on the x86. Proceedings of the USENIX annual technical conference. USENIX '08, Boston, MA, June 22–27, 2008
7. Jensen C, Hagimont D (1998) Protection wrappers a simple and portable sandbox for untrusted applications. Proceedings of the 8th ACM SIGOPS European, SIGOPS European Workshop ACM, Sintra, Portugal, September 7–10, 1998, pp 104–110
8. McCamant S, Morrisett G (2005) Efficient, verifiable binary sandboxing for a CISC srchitecture. Technical Report 2005–030. Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, MIT Laboratory for Computer Science (LCS), MIT, Boston MA, May 2, 2005
9. Song Y, Fleisch BD (2007) Utilizing binary rewriting for improving end-host security. IEEE Transactions on Parallel and Distributed Computing 18(12):1687–1699
10. Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S (2002) Condor-G: a computation management agent for multi-institutional grids. Cluster Comput 5(3):237–246
11. Andreeva J, Campana S, Fanzago F, Herrala J (2002) High-energy physics on the grid: the atlas and CMS experience. J Grid Comput 6(1):3–13

Schemes Based on Rank Codes

Ernst M. Gabidulin
Department of Radio Engineering, Moscow Institute of Physics and Technology (State University), Dolgoprudny, Moscow region, Russia

Synonyms

Applications of rank-metric codes

Related Concepts

►Space-Time Codes

Definition

Schemes based on rank codes appear in many areas of communications, cryptography, and information theory.

Background

The rank function defined on the set of matrices (or vectors) is in fact the norm function. The well-known inequalities for sums of matrices $|\mathrm{Rk}(A) - \mathrm{Rk}(B)| \le \mathrm{Rk}(A+B) \le \mathrm{Rk}(A) + \mathrm{Rk}(B)$ are known from the very beginning of the theory of matrices. They define implicitly the rank distance relations on the space of all matrices of identical size. Nevertheless, the definition and applications of rank-metric-based codes are of active interest only for last decades.

Theory

Optimal Linear Rank Codes

Let $\mathbb{F}_q$ be a base field of q elements and $\mathbb{F}_{q^N}$ be an extension field of degree N. In *vector* representation, rank codes are defined as subsets of a normed n-dimensional space $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$ of n-vectors over an extension field $\mathbb{F}_{q^N}$, where the norm of a vector $\mathbf{v} \in \mathbb{F}_{q^N}^n$ is defined to be the maximal number of coordinates of $\mathbf{v}$ which are linearly independent over the base field $\mathbb{F}_q$. A $\mathbb{F}_{q^N}$-linear vector code $\mathcal{V}$ is a subspace of the normed space $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$. Denote by (n,k,d) a code $\mathcal{V}$ of dimension $k \le n$ and rank distance d. A (n,k,d) code is called optimal, if the Singleton bound $d = n-k+1$ is reached. Optimal codes are also called *maximal rank distance* codes, or, for brevity, MRD codes. Such codes can be described in terms of a full rank *parity-check* matrix $H_{n-k} = H_{d-1}$ over $\mathbb{F}_{q^N}$ of size $(d-1) \times n$.

Let $h_1, h_2, \ldots, h_n$ be a set of elements from the extension field $\mathbb{F}_{q^n}$ *linearly independent* over the base field $\mathbb{F}$. Then, a parity check matrix of the form

$$H_{d-1} = \begin{bmatrix} h_1 & h_2 & \ldots & h_n \\ h_1^q & h_2^q & \ldots & h_n^q \\ h_1^{q^2} & h_2^{q^2} & \ldots & h_n^{q^2} \\ \ldots & \ldots & \ldots & \ldots \\ h_1^{q^{(d-2)}} & h_2^{q^{(d-2)}} & \ldots & h_n^{q^{(d-2)}} \end{bmatrix}$$

defines an MRD (n,k,d) code with code length $n \le N$, dimension $k = n-d+1$ and rank distance $d = n-k+1$.

Equivalently, general constructions of MRD codes can be described in terms of generator matrices. Let $g_1, g_2, \ldots, g_n$ be a set of elements from the extension field $\mathbb{F}_{q^N}$ *linearly independent* over the base field $\mathbb{F}_q$. Then, a generator matrix of the form

$$G_k = \begin{bmatrix} g_1 & g_2 & \ldots & g_n \\ g_1^q & g_2^q & \ldots & g_n^q \\ g_1^{q^2} & g_2^{q^2} & \ldots & g_n^{q^2} \\ \ldots & \ldots & \ldots & \ldots \\ g_1^{q^{(k-1)}} & g_2^{q^{(k-1)}} & \ldots & g_n^{q^{(k-1)}} \end{bmatrix}.$$

defines an MRD (n,k,d) code with code length $n \le N$, dimension $k = n-d+1$, and rank distance $d = n-k+1$.

Code words of a vector code $\mathcal{V}$ are n-vectors over the extension field $\mathbb{F}_{q^N}$. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ be a basis of $\mathbb{F}_{q^N}$ over $\mathbb{F}_q$. Then, each vector $\mathbf{v} = [v_1 \quad v_2 \quad \ldots \quad v_n] \in \mathbb{F}_{q^N}^n$ can be mapped into the $N \times n$ matrix $M \in \mathbb{F}_q^{N \times n}$ by replacing each coordinate v_j with the N-column consisting of coefficients in representing v_j by the basis Ω. This mapping is bijective and isometric.

Given a rank code $\mathcal{V}$ in vector representation, one can construct a rank code $\mathcal{M}$ in matrix representation with the same size, code distance, and pairwise distances, and vice versa.

For applications, it is more convenient to fulfil encoding and decoding in the vector representation. Transmitting is realized usually with the matrix representation:

Encoding
$$\mathbf{u} \xrightarrow[\text{vector representation}]{} \mathbf{g}(\mathbf{u}) = \mathbf{u}G_k \xrightarrow[\text{matrix representation}]{} A(\mathbf{g}(\mathbf{u})) = M(\mathbf{u})$$

Transmission
$$\xrightarrow[\text{matrix representation}]{} M(\mathbf{u})$$

Receiving
$$\xrightarrow[\text{matrix representation}]{} Y = M(\mathbf{u}) + E_{\text{total}}$$

Decoding
$$\xrightarrow[\text{matrix representation}]{} Y = M(\mathbf{u}) + E_{\text{total}} \xrightarrow[\text{vector representation}]{} \mathbf{y} = \mathbf{g}(\mathbf{u}) + \mathbf{e}_{\text{total}} \to \mathbf{g}(\mathbf{u}) \to \mathbf{u}\,.$$

S

Parallel Channels and Correcting Random Rank Errors and Rank Erasures

Let a transmitting code signal be a $N \times n$ matrix M. Each row is transmitted through a separate channel. Each column corresponds to a time slot. The received matrix has the form $Y = M + E_{\text{total}}$, where E_{total} is a matrix of errors. In general, the matrix E_{total} can be represented as a sum of three matrices: $E_{\text{total}} = E_{\text{random}} + E_{\text{row}} + E_{\text{col}}$. Here the matrix $E_{\text{random}} = TU$ has an *unknown* rank t and is the product of an *unknown* full rank $N \times t$ matrix T and an *unknown* full rank $t \times n$ matrix U. This part is called a *random* rank error of rank t. A matrix $E_{\text{row}} = AR$ has the rank v *known* to the decoder and is the product of a full rank $N \times v$ matrix A *known* to the decoder and an *unknown* full rank $v \times n$ matrix R. This part is called a *row* rank erasure of rank v since each row of E_{row} is a linear combination with *known* coefficients of v *unknown* rows. A matrix $E_{\text{col}} = WC$ has the rank l *known* to the decoder and is the product of an *unknown* full rank $N \times l$ matrix W and a full rank $l \times n$ matrix C *known* to the decoder. This part is called a *column* rank erasure of rank l since each column of E_{col} is a linear combination with *known* coefficients of l *unknown* columns.

For parallel channels, the matrix $E_{\text{row}} = AR$ contains $N - v$ *zero rows* with indices known to the decoder. Similarly, the matrix E_{col} contains $n - l$ zero columns with indices known to the decoder.

To correct rank errors and erasures, a received matrix Y must be converted to a vector $\mathbf{y}$ over the extension field $\mathbb{F}_{q^N}$. It follows that $\mathbf{y} = \mathbf{g}(\mathbf{u}) + \mathbf{e}_{\text{total}}$, where

$$\begin{aligned}\mathbf{e}_{\text{total}} &= \mathbf{e}_{\text{random}} + \mathbf{e}_{\text{row}} + \mathbf{e}_{\text{col}}\\ &= e_1\mathbf{u}_1 + e_2\mathbf{u}_2 + \cdots + e_t\mathbf{u}_t +\\ &\quad + a_1\mathbf{r}_1 + a_2\mathbf{r}_2 + \cdots + a_v\mathbf{r}_v +\\ &\quad + w_1\mathbf{c}_1 + w_2\mathbf{c}_2 + \cdots + w_l\mathbf{c}_l.\end{aligned}$$

The part $\mathbf{e}_{\text{random}} = e_1\mathbf{u}_1 + e_2\mathbf{u}_2 + \cdots + e_t\mathbf{u}_t$ is a *vector random* error of rank t under assumption that elements $e_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}_q$ and *unknown* to the decoder; n-vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t$ have coordinates in the *base* field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$ and are *unknown* to the decoder. The rank t is *unknown* to the decoder.

The part $\mathbf{e}_{\text{row}} = a_1\mathbf{r}_1 + a_2\mathbf{r}_2 + \cdots + a_v\mathbf{r}_v$ is a *vector rank* ***row*** *erasure* under assumption that elements $a_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}$ and *known* to the decoder; n-vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_v$ have coordinates in the *base* field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$, and are *unknown* to the decoder.

The part $\mathbf{e}_{\text{col}} = w_1\mathbf{c}_1 + w_2\mathbf{c}_2 + \cdots + w_l\mathbf{c}_l$ is a *vector rank* ***column*** *erasure* under assumption that elements $w_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}_q$ and are *unknown* to the decoder; n-vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_l$ have coordinates in the base field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$, and are *known* to the decoder.

For MRD codes, correcting rank errors and rank erasures is fulfilled by calculating the syndrome $\mathbf{s} = \mathbf{y}H_{d-1}^{\top} = \mathbf{e}_{\text{total}}H_{d-1}^{\top}$. The total error $\mathbf{e}_{\text{total}}$ can be found uniquely from $\mathbf{s}$ provided that $2t + v + l \le d - 1$. There exist several fast algorithms to correct either random rank errors only, or both random rank errors and row and column erasures simultaneously [1–4].

Rank Codes as Space-Time Codes

Space-time codes are introduced in [5]. The model for a communication system in brief is as follows. The transmitting side is equipped with N transmit antennas. A signal is transmitted in time slots $1, 2, \ldots, n$. The set of all possible signals to transmit is the set $\mathcal{C} = \mathbb{A}^{N\times n}$ of all the $N \times n$ matrices with entries in the set $\mathbb{A}$. The set $\mathbb{A}$ of size q is a set of chosen complex numbers.

A space-time code (STC) $\mathcal{M}$ is any set of matrices from $\mathcal{C}$. The design criterion of a STC is as follows: *Choose a code $\mathcal{M}$ in such a manner that the difference $M_i - M_j$, $i \ne j$, has full rank.*

MRD (n, k, d) codes over finite fields with rank distance $d = n$ and $k = 1$ satisfy the above criterion. Nevertheless, these codes cannot be used directly as space-time codes since processing of received signals is carried out in the complex field. However, it is still possible to use *MRD* codes over finite fields as *templates*. For example, assume that $\mathbb{A} = \{a, b\}$, where a, b are different complex numbers. First, construct a MRD code with rank distance $d = n$ consisting of $N \times n$ matrices over the field $\mathbb{F}_2 = \{0, 1\}$. Next, replace in all matrices "0" 's by "a" 's and "1" 's by "b" 's. The resulting code is a code over the complex field satisfying the design criterion. Similar results are obtained for other finite fields [6].

Rank Codes in Network Coding

Consider a communication network, where a single source transmits information to a single destination. The model of a network was proposed and investigated in [4]. The source formats the information to be transmitted into N packets $X(1), \ldots, X(N)$ of length $N + n$ over the finite field $\mathbb{F}_q$ and constructs a $(N \times (N + n))$ matrix X with these packets as rows. The source choose a code $\mathcal{X}$ consisting of matrices X, which can be transmitted.

Each intermediate node calculates random linear combinations of ingoing packets, where a packet is represented as an element of a finite field $\mathbb{F}_{q^{N+n}}$. The node retransmits

randomly calculated packets. Therefore, the destination collects a random number N_r of packets $Y(1), \ldots, Y(N_r)$ of length $N + n$ and creates a $N_r \times (N + n)$ matrix Y.

The problem is to recover the original packets $X(1), \ldots, X(N)$, or the matrix X from the received matrix Y.

The basic model of a channel induced by random network coding is described as follows. The transmitted matrix X and the received matrix Y are connected by the relation $Y = AX + BZ$, where A is an $N_r \times N$ matrix corresponding to the overall linear transformation applied by intermediate nodes of the network; Z is an $l \times (N+n)$ matrix whose rows are the error packets $z_1, \ldots, z_l$; B is an $N_r \times l$ matrix corresponding to the overall linear transformation applied to $z_1, \ldots, z_l$ on route to the destination. The number of nonzero rows of Z gives the total number of corrupt packets injected in the network. Random matrices A, B, Z are unknown to the destination.

It is proposed in [4] to apply so-called lifting construction for constructing a code $\mathcal{X}$. Each matrix $X \in \mathcal{X}$ has the form $X = [I_N \quad M]$, where I_N is the identity matrix of order N while $M \in \mathcal{M}$ is a code matrix of some matrix code $\mathcal{M}$ consisting of $N \times n$ matrices over the field $\mathbb{F}_q$. A code $\mathcal{M}$ is assumed to be a MRD *rank code* with rank distance d, if $n \le N$, or a transposed MRD rank code, if $N < n$. The corresponding code $\mathcal{X}$ were analyzed in [4]. Decoding codes $\mathcal{X}$ can be reduced to decoding embedded rank codes $\mathcal{M}$. If the matrix $X = [I_N \quad M]$ is transmitted, then the matrix $Y = AX + BZ = [A + BZ_1 \quad AM + BZ_2]$ is received with unknown to the destination matrices A, B, Z_1, Z_2. By a linear transformation of rows and injecting all zero rows, the part $[A + BZ_1]$ can be reduced to the upper triangular matrix of order N. Elements of the main diagonal are "0"'s or "1"'s. The number of "1"'s is equal to the rank of $[A + BZ_1]$. The same operations over the matrix $[AM + BZ_2]$ allows to extract the submatrix of the form $R = M + LM + DC$, where R, L, and C are known matrices. Thus, the result is a matrix M of the rank code $\mathcal{M}$ corrupted by a *row* rank erasure LM and a *column* rank erasure DC. The unknown matrix M can be uniquely recovered from R provided that $\mathrm{Rk}(L) + \mathrm{Rk}(C) \le d - 1$.

Other network codes are known generalizing constructions above (see [7]).

Public Key Cryptosystems Based on Rank Codes

Public key cryptosystems (PKC) based on linear codes were proposed in [8]. Rank code–based PKC (insecure) was described in [9]. The variant withstanding known attacks can be found in [10]. The simplest case follows. The cryptographer chooses as secret keys a generator matrix G_k of a *maximal rank distance* (MRD) $(n, k, d = n - k + 1)$ code, a square matrix S of order k over the extension field $\mathbb{F}_{q^N}$, a square matrix P of order n with special properties. Matrices S and P are called the row scrambler and the column scrambler, correspondingly. The product of three matrices $G_{\mathrm{pub}} = SG_kP$ is declared as the *public* key. Plaintexts are vectors $\mathbf{m} = [m_1 \quad m_2 \quad \ldots \quad m_k]$. Given a plaintext $\mathbf{m}$, a ciphertext is calculated as $\mathbf{c} = \mathbf{m}G_{pub} + \mathbf{e}$, where $\mathbf{e}$ is a random artificial error. To decrypt a ciphertext $\mathbf{c}$, the legitimate party upon receiving $\mathbf{c}$ calculates the intermediate ciphertext $\widetilde{\mathbf{c}} = \mathbf{m}SG_k + \mathbf{e}P^{-1}$. The parameters P and $\mathbf{e}$ should be chosen in such a manner that the user can recover easily the plaintext $\mathbf{m}$ from $\widetilde{\mathbf{c}}$. The proper choice of artificial errors $\mathbf{e}$ is as follows. Represent $\mathbf{e}$ as $\mathbf{e} = [\mathbf{e}_1 \quad \mathbf{e}_2]$, where length of $\mathbf{e}_1$ is equal to a. Choose this part randomly. Choose $\mathbf{e}_2$ such that $\mathrm{Rk}(\mathbf{e}_2 \mid \mathbb{F}_q) = b$. The integers a and b must satisfy the condition $2a + b \le (d-1)/2$. To choose a suitable matrix P, represent a matrix P^{-1} in the block form as $P^{-1} = \left[\begin{smallmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{smallmatrix}\right]$, where $\mathbf{q}_{11}$ is the square matrix of order a, $\mathbf{q}_{21}$ is the matrix of size $a \times (n - a)$, $\mathbf{q}_{12}$ – of size $a \times (n - a)$, and $\mathbf{q}_{22}$ is the square matrix of order $(n - a)$. Choose entries of matrices q_{11}, q_{21}, q_{12} in the *extension field* $\mathbb{F}_{q^N}$ but with the condition for the column rank $\mathrm{Rk}(q_{12} \mid \mathbb{F}_q) \le a$. Choose entries of the matrix q_{22} in the *base field* $\mathbb{F}_q$. Under these conditions, the rank of the error $\mathbf{e}P^{-1}$ in the intermediate ciphertext $\widetilde{\mathbf{c}}$ is not greater than $2a + b \le (d-1)/2$. Hence, the legitimate party can extract from $\widetilde{\mathbf{c}}$ by using a fast-decoding algorithm the scrambled plaintext $S\mathbf{m}$ and then get the plaintext as $S^{-1}S\mathbf{m} = \mathbf{m}$.

Open Problems

Some open problems are: correcting rank errors beyond the $(d - 1)/2$ bound; isometric mapping maximal rank distance codes into signals over specific complex constellations; simultaneous data protection and error correction; network coding on the base of rank codes; designing new rank-metric-related public key cryptosystems.

Recommended Reading

1. Gabidulin EM (1985) Theory of codes with maximum rank distance. Probl Inf Transm 21(1):1–12
2. Gabidulin EM, Paramonov AV, Tretjakov OV (1992) Rank errors and rank erasures correction. In: Proceedings of the 4th international colloquium on coding theory, 30 September–7 October 1991, Dilijan, Armenia, pp 11–19, Yerevan, 1992
3. Gabidulin EM, Pilipchuk NI (2008) Error and erasure correcting algorithms for rank codes. Designs Codes and Cryptogr 49:105–122. DOI 10.1007/s10623-008-9185-7
4. Silva D, Kschischang FR, Koetter R (2008) A rank-metric approach to error control in random network coding. IEEE Trans Inf Theory 54(9):3951–3967
5. Tarokh V, Jafarkhani H, Calderbank AR (1998) Space-time codes for high data rate wireless communication: performance

criterion and code construction. IEEE Trans Inf Theory 44(2):744–765
6. Gabidulin EM, Lusina P, Bossert M (2003) Maximum rank codes as space-time codes. IEEE Trans Inf Theory 46(10):2757–2760
7. Gabidulin EM, Bossert M (2009) Algebraic codes in network coding. Probl. Inf Transm 45(4):3–17
8. McEliece RJ (1978) A public key cryptosystem based on algebraic coding theory. JPL DSN progress report 42–44, Pasadena, pp 114–116
9. Gabidulin EM, Paramonov AV, Tretjakov OV (1991) Ideals over a non-commutative ring and their application in cryptology. In: Davies DW (ed) Advances in cryptology | Eurocrypt '91. Lecture notes in computer science, vol 547. Springer, Berlin and Heidelberg, pp 482–489
10. Gabidulin EM (2008) Attacks and counter-attacks on the GPT public key cryptosystem. Designs Codes Cryptogr 48(2):171–177. DOI 10.1007/s10623-007-9160-8

Schnorr Digital Signature Scheme

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Digital Signature Scheme; ►Digital Signature Standard; ►ElGamal Digital Signature Scheme

Background

The Schnorr signature scheme [6] is derived from ►Schnorr's identification protocol using the Fiat–Shamir heuristic [2]. The resulting ►digital signature scheme is related to the ►Digital Signature Standard (DSS). As in DSS, the system works in a subgroup of the ►group $\mathbb{Z}_p^*$ for some ►prime number p. The resulting signatures have the same length as DSS signatures.

Theory

The signature scheme works as follows:

Key Generation. Same as in the DSS system. Given two security parameters τ, $\lambda \in \mathbb{Z}\,(\tau > \lambda)$ as input do the following:

1. Generate a random λ-bit prime q.
2. Generate a random τ-bit prime p such that q divides $p-1$.
3. Pick an element $g \in \mathbb{Z}_p^*$ of order q.
4. Pick a random integer $\alpha \in [1, q]$ and compute $y = g^{\alpha} \in \mathbb{Z}_p^*$.
5. Let H be a ►hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.

The resulting public key is (p, q, g, y, H). The private key is (p, q, g, α, H) do:

Signing. To sign a message $m \in \{0,1\}^*$ using the private key (p, q, g, α, H) do:

1. Pick a random $k \in \mathbb{Z}_p^*$.
2. Compute $r = g^k \in \mathbb{Z}_p^*$. Set $c = H(m \,\|\, r) \in \mathbb{Z}_q$ and $s = \alpha c + k \in \mathbb{Z}_q$.
3. Output the pair $(s, c) \in \mathbb{Z}_q^2$ as the signature on m.

Verifying. To verify a message/signature pair $(m, (s, c))$ using the public key (p, q, g, y, H) do:

1. Compute $v = g^s y^{-c} \in \mathbb{Z}_p$.
2. Accept the signature if $c = H(m \,\|\, v)$. Otherwise, reject.

We first check that the verification algorithm accepts all valid message/signature pairs. For a valid message/signature pair we have

$$v = g^s\, y^{-c} = g^{\alpha c + k}\, y^{-c} = \left(y^c g^k\right) y^{-c} = g^k \in \mathbb{Z}_p$$

and therefore $H(m \,\|\, v) = H(m \,\|\, g^k) = c$. It follows that a valid message/signature is always accepted.

The signature can be shown to be existentially unforgeable (►existential forgery) under a chosen message attack in the ►random oracle model, assuming the ►discrete logarithm problem in the group generated by g is intractable. This proof of security is a special case of a general result that shows how to convert a public-coin authentication protocol (a protocol in which the verifier only contributes randomness) into a secure signature scheme in the random oracle model [1, 5]. In the proof of security, the function H is assumed to be a random oracle. In practice, one derives H from some cryptographic has function such as ►SHA-1.

To discuss signature length, we fix concrete security parameters. At the present time, the discrete-log problem in the cyclic group $\mathbb{Z}_p^*$ where p is a 1024-bit prime is considered intractable [3] except for a very well-funded organization. Schnorr signatures use a subgroup of order q of $\mathbb{Z}_p^*$. When q is a 160-bit prime, the discrete log problem in this subgroup is believed to be as hard as discrete-log in all of $\mathbb{Z}_p^*$, although proving this is currently an open problem. Hence, for the present discussion we assume p is a 1024-bit prime and q is a 160-bit prime. Since a Schnorr signature contains two elements in $\mathbb{Z}_q$ we see that, with these parameters, its length is 320-bits.

Schnorr signatures are efficient and practical. The time to compute a signature is dominated by one exponentiation and this exponentiation can be done off-line, i.e., before the message is given. Verifying a signature is dominated by the time to compute a multi-exponentiation of the form $g^a h^b$ for some $g, h \in \mathbb{Z}_p$ and $a, b \in \mathbb{Z}_q$. Multi-exponentiations of

this type can be done at approximately the cost of a single exponentiation [4, p. 617].

Recommended Reading

1. Abdalla M, An J, Bellare M, Namprempre C (2002) From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2004. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 418–433
2. Fiat A, Shamir A (1986) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in cryptology – CRYPTO'86. Lecture notes in computer science, vol 263. Springer, Berlin, pp 186–194
3. Lenstra A, Verheul E (2001) Selecting cryptographic key sizes. J Cryptol 14(4):255–293
4. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
5. Ohta K, Okamoto T (1998) On concrete security treatment of signatures derived from identification. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 354–369
6. Schnorr C (1991) Efficient signature generation by smart cards. J Cryptol 4(3):161–174

Schnorr Identification Protocol

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Related Concepts

►Authentication; ►Entity Authentication; ►Identification

Definition

The Schnorr Identification Protocol is a public-key based *challenge-response identification* protocol.

Background

The Schnorr Identification Protocol was introduced after, and is comparable to, the identification protocol of Fiat and Shamir and that of Guillou-Quisquater. Schnorr's protocol relies upon the security of the *Discrete Logarithm Problem*.

Theory

Schnorr's protocol is quite simple to describe, and makes novel use of two submitted messages in order to reduce computational complexity. The first message, called a *witness*, does not depend on either a secret value or a challenge value. The second message, more typically a response in a *challenge-response identification* protocol, is a function of both a secret value and a challenge value, but also of the original witness. The witness can be pre-computed and stored, reducing the computational complexity that might otherwise be required at the time of the protocol.

As a setup for the Schnorr protocol, the following system-wide parameters are chosen, and assumed to be accessible to all users:

- A *prime number* r and suitable prime q such that $q|r-1$
- A *generator* $1 \leq \alpha \leq p-1$ such that $\alpha^{r-1} \equiv 1 \pmod{r}$, from which the value β of multiplicative order q may be computed as $\beta = \alpha^{(r-1)/q} \bmod r$

(r, q, β) are then published and available to all participants. Each user additionally has a private and public key (►Public Key Cryptography).

With this setup, the protocol proceeds as follows with the prover (P) trying to successfully identify to the verifier (V), where P has the private key $0 \leq p \leq q-1$ and V has access to a trusted copy of P's corresponding public key $y_P = \beta^{-p} \bmod r$.

1. P (pre-)computes a so-called *witness* $w = \beta^m \bmod r$, where the *commitment* $1 \leq m \leq q-1$ is chosen randomly. P sends w to V.
2. Upon receipt of w, V sends a challenge c to P, where $1 \leq c \leq 2^t - 1$ for appropriate security parameter t (Schnorr originally suggests $t = 72$).
3. Upon receipt of the challenge c, P computes the *response* $s = pc + m \bmod q$ to V.
4. Upon receipt of the response s, V computes $z = \beta^s y_P{}^c \bmod r$ and accepts P's identity if $z = w$.

Note that the logical steps above can be compressed into a single exchange from P to V. The security of Schnorr-like identification schemes was shown by Bellare, Namprepre, and Neven.

Applications

Schnorr's Identification Protocol was originally intended for constrained application environments, and thus has relatively low computational complexity resulting from computations in the subgroup of size q, and supports a pre-computation of the *witness*.

Recommended Reading

1. Bellare M, Namprempre C, Neven G (2009) Security proofs for identity-based identification and signature schemes. J Cryptol 22(1):1–61
2. Claus-Peter Schnorr (1989) Efficient identification and signatures for smart cards. In: Brassard G (ed) CRYPTO, Lecture notes in computer science, vol 435. Springer, Berlin, pp 239–252
3. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton

Script Language Security

Claudio A. Ardagna, Ernesto Damiani
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Security of web browser scripting languages

Related Concepts

►Sandbox Security Model; ►Static Code Analysis

Definition

Script language security refers to all activities needed to secure the execution of script-based applications, downloaded from a remote host and executed on a user's local machine. Script-based applications may contain malicious software that attacks the local machine and causes information leakage and/or data losses.

Background

The increasing number of reports of security-relevant software faults shows that the problem of verifying security-related properties needs to be addressed, especially in the development of high-integrity systems where safety and security are paramount. Software verification techniques have been categorized in three broad categories involving (i) test-based verification; (ii) model checking, for formal verification of software systems with respect to specifications expressed in a logical framework; and (iii) static code analysis, for retrieving valuable information about a program by analyzing its code. In addition, some techniques for run-time checking have been proposed to monitor software execution and potentially identify misbehaving. In this framework, the security community has considered the problem of protecting software systems from attacks brought on by mobile code, as well as by server- and client-side scripts. Nowadays, script languages (e.g., JavaScript) are widely adopted, especially in Web application development, and represent an important component in several aspects of the global communication infrastructure. However, script security poses different problems from traditional code security and needs ad-hoc solutions. Some malicious client-side scripts try and collect information on the client environment and/or disrupt client machine. Some of these attacks and potential countermeasures will be discussed in the next sections, focusing on the security model introduced by the deployment of Java technology.

Theory

Mobile code is transferred from a remote host and executed on a user's local system. Often, it is embedded in other file formats, as for instance JavaScript within HTML pages. Attacks have been categorized in four classes [7]: (i) *system modification*, the most critical, when the script tries to modify the host system getting access to critical data; (ii) *invasion of privacy*, when the script tries to breach the user's privacy by disclosing personal information of the user or of the hosting machine; (iii) *denial of service*, when the script tries to make resources unavailable to other users who rightfully try to access them; (iv) *antagonism*, the less critical, when the script just tries to annoy the user in some way.

Scripts are written in high-level programming languages that are interpreted (rather than compiled) and executed within a Web browser or a Web server. Most script languages include advanced library support for accessing resources in the local environment or in the remote environment they came from. Today, script languages are massively adopted in the development of Web sites. When a user accesses a Web page containing a script, the script is automatically downloaded on the user's local machine and interpreted within the browser with little or few checks on the security side effects of the execution. To alleviate this problem, a first approach is based on defining security policies that restrict the script access to remote or local files and resources. A typical restriction prevents a script to connect back to any other Web server but the one it was downloaded from. The term *namespace* is often used to designate the hierarchy of objects accessible by a script in a given context.

Malicious scripts may result in serious security flaws. An attacker can exploit Web site vulnerabilities to inject malicious client-side scripts in a Web page, thus collecting personal information of the users accessing such a page. Such attacks (►Cross-Site Scripting – XSS for details) may result in (i) *data loss*, when data provided by the user in a communication are accessed by someone who is not the intended recipient, or (ii) *privacy loss*, when private data of the users are disclosed without the explicit consent of the users [1]. XSS attacks have been reported since 1990, and are among the most frequently reported vulnerabilities [3]. As an example of attack, malicious scripts can open HTTP connections and send back to the attacker personal information of the users without any access to the operating systems or underlying applications. A malicious script can access personal data typed in forms, such as credit card numbers, passwords, cookies, and many others. Since data are directly accessed in the browser, no secure communication technique can be used to protect the user [1]; rather, to counteract XSS attacks, the Web site developers have to implement strong applications that carefully check and filter input data, thus avoiding script

injection. If this is not the case, the user can only protect itself by either disabling the scripts or by accessing trusted sites only.

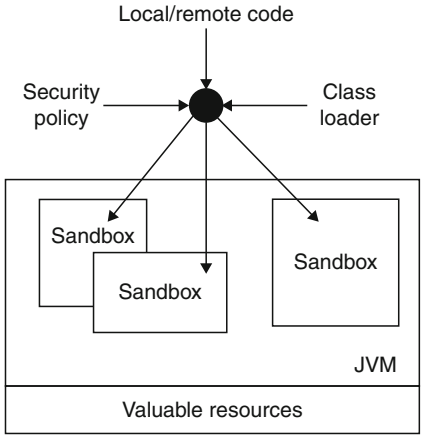
Java Security Model

Since its advent, Java programming language has had a great impact and success in the development of both traditional stand-alone applications and mobile code software (e.g., Java Applet and JavaScript). To manage security and counteract attacks to Java-based applications, the Java Security Model aims to [5]:

- Provide the Java platform as a secure, ready-built platform on which to run Java-enabled applications in a secure fashion.
- Provide security tools and services implemented in the Java programming language that enable a wider range of security-sensitive applications, for example, in the enterprise world.

Java security provides different levels of protection ranging from language-level security, to Java Virtual Machine (JVM) level security, and Java security API [2]. First, the language-level security derives from the basic characteristics of the language, that provides a simplified and easy-to-use solution, strictly object-oriented, strongly typed, and with no possibility of using pointers and address arithmetic. Then, the JVM, as the component responsible for the execution of applications, implements some important security features, such as bytecode verification, runtime safety checks, access control and permission evaluation and enforcement. Moreover, Java Security API provides a set of classes that can be used to secure Java applications. Among them, Java Cryptography Architecture (JCA) for cryptographic capabilities, Java Authentication and Authorization Service (JAAS) for user's authentication and authorization, and Java Secure Socket Extension (JSSE) for secure communication stand out.

Another important part of the Java security model is based on the concept of *sandbox*. Originally, to secure Java applications and their execution, the Java sandbox model distinguished between local code that was executed by the JVM with full access to the system resources, and remote code (e.g., scripts) obtained by the open network that was executed in a strict environment (i.e., sandbox) with very limited access to the system resources. With the introduction of the JDK1.1, the sandbox model has been refined, introducing a separation between trusted and untrusted remote code. Trusted code represented code signed by a trusted authority and executed as a local application, while untrusted code was still executed in the sandbox. Finally, the current sandbox model has been introduced by the



Script Language Security. Fig. 1 Java 2 Platform sandbox model [5]

Java 2 Platform and is shown in Fig. 1. This model departs from the concept of trusted code used by previous models, and introduces a finer-grained access control with permissions associated to each application. The Java 2 sandbox also provides easily configurable security policy to govern what an application can do, easily extensible access control structure, and extension of security checks to all Java programs. In general, the sandbox approach provides a suitable environment for executing untrusted scripts in a strict environment with limited access to resources, thus reducing the effects on malicious scripts.

Although, the sandbox model represents an important means to protect the users by malicious mobile code, it may result limiting in some cases. As a consequence, in the past, some techniques for security formal modeling and verification of Java bytecodes have been provided by means of theorem proving and model checking (e.g., [6]). Also, some solutions introduced the concept of model-carrying code to ensure security of mobile untrusted code (e.g., [8]). These approaches enable a user to understand and formally reason about what a piece of mobile code can do, and to check if the code fulfills its security policies before executing the application. Finally, some solutions (e.g., [9]) applied static detection of security vulnerabilities to scripting languages, while others approaches (e.g., [10]) could track the flow of untrusted data through a program.

Open Problems

Static code analysis and formal methods have been applied to program verification since long and today are routinely applied to large code bases, such as an entire Linux distribution [4]. A major drawback of these approaches is performance, since code analysis may require a huge

amount of time. Scripts however are intrinsically shorter and simpler; static analysis techniques can then become a key solution to provide runtime analysis of code before its execution on the local machine. In this realm, the research community is moving fast and also some applications have been provided to monitor script execution, such as, Firebug (http://getfirebug.com).

An important aspect toward securing script language execution is the integration of traditional solutions, like the Java sandbox, with formal solutions, such as static code analysis. A hybrid solution may perform an a priori analysis first, and then apply sandbox restrictions. This hybrid solution may balance between the amount of static code analysis performed and the amount of restrictions applied in the sandbox. In other words, the more complete the static analysis of the code, the less is the amount of restrictions to be applied in the sandbox. This hybrid solution must be developed to identify intelligent malicious code that tries to confuse an analyzer, for instance, by replacing the character @ of the mail at which personal information must be sent back with the related ASCII value.

Recommended Reading

1. Anupam V, Mayer A (1998) Security of web browser scripting languages: vulnerabilities, attacks, and remedies. In: Proceedings of the 7th Conference on USENIX Security Symposium, San Antonio, Texas, January 1998
2. Cholakov N, Milev D (2005) The evolution of the java security model. In: Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech 2005), Varna, Bulgaria, June 2005
3. Christey S, Martin RA (2007) Vulnerability type distributions in CVE (version 1.1), MITRE Corporation, May 2007. http://cwe.mitre.org/documents/vuln-trends/index.html
4. Damiani E, Ardagna CA, El Ioini N (2009) Open source systems security certification. Springer, New York
5. Gong L. JavaTM 2 Platform Security Architecture, vol. 2. http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-spec.doc1.html
6. Leroy X (2003) Java bytecode verification: algorithms and formalizations. J Autom Reasoning 30(3–4):235–269
7. McGraw G, Felten E (1999) Securing JAVA: getting down to business with mobile code. John Wiley & Sons, Inc, New York
8. Sekar R, Venkatakrishnan VN, Basu S, Bhatkarand S, DuVarney DC (2003) Model-carrying code: a practical approach for safe execution of untrusted applications. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, NY, October 2003
9. Xie Y, Aiken A (2006) Static detection of security vulnerabilities in scripting languages. In: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, Canada, July 2006
10. Xu W, Bhatkar S, Sekar R (2006) Taint-enhanced policy enforcement: a practical approach to defeat a wide range of attacks. In: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, Canada, July 2006

SEAL

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI, CNRS/Lab-STICC/CID and Telecom Bretagne, Brest Cedex 3, France

Synonyms

Software-optimized encryption algorithm

Related Concepts

►Synchronous Stream Cipher

Definition

SEAL stands for Software-optimized Encryption ALgorithm. It is a binary additive stream cipher (Synchronous Stream Ciphers).

Background

SEAL has been proposed in 1993, and several versions have been published: SEAL 1.0, SEAL 2.0 and SEAL 3.0 [1, 2].

Applications

Some attacks have been published that show how SEAL 1.0, SEAL 2.0 [3], and later SEAL 3.0 [4] can be distinguished from a true random function. But there is no really practical attack for the moment.

SEAL has been designed to be really efficient in its software implementation, mainly for 32-bit processors. It is a length-increasing pseudorandom function that maps a 32-bit sequence number n to an L-bit keystream, under control of a 160-bit secret key. A more precise description can be found in the original papers. Note that SEAL 3.0 is covered by two patents in the USA [5].

Recommended Reading

1. Rogaway P, Coppersmith D (1994) A software-optimized encryption algorithm, FSE'94. Lecture notes in computer science, vol 809. Springer, pp 56–63
2. Rogaway P, Coppersmith D (1998) A software-optimized encryption algorithm. J Cryptol 11(4):273–287
3. Handschuh H, Gilbert H (1997) χ^2-Cryptanalysis of the SEAL encryption algorithm, FSE'97. Lecture notes in computer science. Springer, pp 1–12
4. Fluhrer SR (2001) Cryptanalysis of the SEAL 3.0 Pseudorandom Function Family, FSE'01. Lecture notes in computer science, vol 2355. Springer, pp 135–143
5. US Patent 5,454,039 Software-efficient pseudorandom function and the use thereof for encryption; US Patent 5,675,652 Computer readable device implementing a software-efficient pseudorandom function encryption

Sealed Storage

Paul England
Microsoft Corporation, Redmond, WA, USA

Related Concepts

►Attestation; ►Trusted Platform Module (TPM)

Definition

Sealed Storage is a cryptographic data protection facility that is conventionally implemented using a pair of operations called Seal (used to protect data) and Unseal (used to unprotect data). It differs from other symmetric and asymmetric encryption primitives in that the sealer can specify limitations on the software environments that can access the data. The Sealed Storage facility will check that the identity of the requesting software meets policy specified in an earlier Seal operation before the protected data is revealed.

Sealed Storage in the TPM

The primary realization of the Sealed Storage concept is in the ►Trusted Platform Module (TPM) cryptographic processor [1].

TPMs are designed to be incorporated into platforms with firmware that provides the TPM with a reliable report of the system software that is booting on the platform. System software may extend the report by measuring and reporting additional OS components as they are loaded. The TPM protects and maintains this extend-only log of loaded components as a cumulative hash in internal registers called Platform Configuration Registers, or PCRs. [2]

Platform firmware typically uses very simple software measurement primitives like the hash of the software loaded. Upper level software typically reports more complex policies to the TPM and ensures that the components meet the reported policy.

If a computer follows this procedure as it boots, then at any time, the values in the PCR registers represent the software running on the platform earlier in boot. This does not necessarily mean that the PCRs are a useful representation of the security properties of the executing software because upper-level software may load and run untrustworthy components. However, if the system software forms a Trusted Computing Base that can defend itself, and if the system software measures and reports any program or configuration change that changes the security properties of the trusted computing base, then the PCR registers are a useful representation of the Trusted Computing Base executing on the platform. This is the scenario supported by Sealed Storage.

The TPM provides several authentication and authorization primitives that use PCRs. ►Attestation is cryptographic reporting of the software configuration to remote parties. This is supported on the TPM with the Quote() operation, which signs the current PCR settings (and externally provided data to provide freshness or for other purposes).

The TPM implementation of sealed storage is exposed through the operations Seal and Unseal. The Seal command is an encryption operation based on a key that is protected by the TPM. The sealer specifies the data to be protected as well as the PCR settings that must be in place for the data to be revealed. The Seal operation produces ciphertext through an authenticated encryption operation of the concatenation of the data to be protected and the PCRs required.

Later, the same or another program calls Unseal using the blob returned from the prior Seal operation. Internally, the TPM decrypts and checks the integrity of the data, and then checks the current PCR values against those specified in the sealed blob. If the data integrity check succeeds and the PCRs are correct, then the protected data is revealed to the caller; otherwise, an error is returned.

The TPM implementation of the Unseal operation also provides source-authentication in the form of the PCR configuration when the Seal operation was performed.

The TPM also supports a public key–based version of Unseal that allows a remote caller to protect data. In this case, any entity with knowledge of the TPM public key can create an encrypted blob containing the required PCR settings and the protected data. The TPM can decrypt the blob but will only reveal the data if the PCRs are set correctly. The TPM command that supports this operation is called ActivateIdentity(). Of course, this operation does not provide seal-time PCR reporting.

Applications

Microsoft's Bitlocker™ hard disk encryption facility includes an option to protect all or part of the hard disk encryption key using the TPM Seal operation. The security goal is to protect the hard disk contents when the data is at rest and the OS is not running. Attacks to be considered include moving the hard disk to another machine and booting a different OS that does not honor the file system security setting of the original OS.

This is achieved by splitting the disk into two partitions: a main encrypted partition that contains the data to be protected, and a small partition that contains plaintext

early-boot code. During setup, a random disk encryption key is generated and the main partition is encrypted. The disk key (or part of the key) is then Sealed to the PCR configuration measurements expected for the early OS boot code on the smaller boot partition. On reboot, the early-boot component attempts to Unseal the disk key and if successful, uses the key to decrypt/encrypt IO requests to the rest of the disk.

If the disk is moved to another platform, the original TPM is not present so the data cannot be Unsealed. If an alternative OS loader is booted, then the PCR configuration will not be correct and the Unseal operation will fail.

Open Problems

Both Attestation and Sealed Storage suffer from a tension between detailed measurement and reporting of the software and hardware configurations, and the problems that arise if configurations change unexpectedly or change too often rendering previously sealed data inaccessible. The TPM implementation of sealed storage allows a configuration with access to a secret to Seal that secret to a future known configuration, but this mechanism does not work if the configuration change was unexpected and previously sealed data is not accessible. Solutions to this problem will either involve reporting more complex policies than code-hashes to the TPM (so that expected configuration changes do not result in data becoming inaccessible), or changing the Sealed Storage implementation to allow more complex access control policies. The latter is being considered for inclusion in the next version of the TPM.

Recommended Reading

Specification

1. TPM main specification, http://www.trustedcomputinggroup.org/resources/tpm_main_specification

Book

2. Pearson S (ed) (2003) Trusted computing platforms: TCPA technology in context. Prentice Hall, New York

Search over Encrypted Data

Ali Bagherzandi, Bijit Hore, Sharad Mehrotra
Donald Bren School of Computer Science, University of California, Irvine, CA, USA

Related Concepts

►Encrypted Data; ►ε-Privacy; ►Trade-off

Definition

Search on encrypted data refers to the ability to identify and retrieve a set of objects from an encrypted collection that satisfy the query without decrypting the data.

Background

Search over encrypted data has been primarily studied in the context of data outsourcing where a client stores potentially confidential information on a remote server that may reside in an "untrusted" environment. The client employs encryption to protect the sensitive contents in the data. Typically, a "curious intruder" model is assumed, wherein the attacker is assumed to be curious about the content of the data, but is not malicious, so as to alter the data itself. The problem is then, how can the client issue queries which the server can evaluate against the encrypted representation of the data and return only the matching records to the client. Techniques to support search over encrypted data function are based on storing additional information (i.e., metadata) which by itself does not leak information about the data but enables the predicates to be evaluated over the encrypted representation. Existing techniques depend upon the nature of data being stored as well as the nature of search queries.

Theory and Applications

A variety of cryptographic and non-cryptographic techniques have been developed to support query-based retrieval over encrypted data. They can be classified into the following two categories:

- ***Keyword-based search on encrypted text documents***: There is a document repository where each document is modeled as a set of keywords from a given dictionary. The keywords are encrypted before storing on the server. The client can then query the server by specifying one or more keywords, in response to which the server returns documents that contain the specified keywords.
- ***Range queries over multidimensional data***: In this setting, the client stores relational/multi-attribute data where one or more attributes may correspond to ordered domains. Queries consist of range search over such multidimensional data.

While the details of the techniques to support retrieval in the two settings differ, the solutions are based on storing additional information in the form of metadata along with the encrypted data representation to allow for corresponding predicates to be evaluated over the metadata. Techniques developed range from cryptographic solutions that attempt to ensure zero leakage of information from the

encrypted representation (albeit, at a higher cost) to non-cryptographic solutions that attempt to explore a trade-off between confidentiality achieved and the overhead of the approach. Some of the well-known approaches are described in the remainder of this entry.

Keyword Search Over Encrypted Documents

Symmetric Key Approach Based on Trapdoors

Amongst the first techniques to support search over encrypted documents is one that uses a symmetric key-based encryption scheme. The solution works as follows:

Consider a data owner, Alice, who wishes to store a collection of documents ($\mathcal{D}$) with Bob (the server). Alice encrypts each document $D \in \mathcal{D}$ prior to storing it with Bob. In addition, Alice creates a "secure index", $I(D)$, which is stored at the service provider that will help her perform keyword search. The secure index is such that it reveals no information about its content to Bob. However, it allows him to test for presence or absence of keywords using a *trapdoor* associated with the keyword, where a trapdoor is generated with a secret key that resides with Alice. When Alice wants to search for documents containing word w, she generates a trapdoor for w which can then be used by Bob to retrieve (and return) the relevant documents.

The secure index is created over the keywords in D as follows. Let document D consist of the sequence of equal length words $w_1, \ldots, w_l$. (The words can be made equi-length by padding wherever necessary). Alice pre-encrypts each word using a deterministic encryption algorithm E_{k_p}, where the key k_p is kept private. The document is now represented as a sequence of E-encrypted words $x_1, \ldots, x_l$. The index is created by computing the bitwise XOR (denoted by the symbol $\oplus$) of the encrypted-text with a sequence of pseudorandom bits $s_1, \ldots, s_l$ that Alice generates using a stream cipher. Each s_i is $n - m$ bit long and for each pseudorandom string s_i, Alice computes a pseudorandom function $F_{k_c}(s_i)$ (defined below) seeded on key k_c which generates a random m-bit sequence. Using the result of $F_{k_c}(s_i)$, Alice computes a n-bit sequence $t_i :=< s_i, F_{k_c}(s_i) >$, where $< a, b >$ denotes concatenation of the string a and b. Now to encrypt the n-bit word x_i, Alice computes the XOR of x_i with t_i, i.e., ciphertext $c_i := x_i \oplus t_i$. Since only Alice can generate the pseudorandom stream $t_1, \ldots, t_l$, no one else can decrypt c_i.

Pseudorandom Functions

A pseudorandom function denoted as $F : K_F \times X \rightarrow Y$, where K_F is the set of keys, X denotes the set $\{0,1\}^n$ and Y denotes the set $\{0,1\}^m$. Intuitively, a pseudorandom function is computationally indistinguishable from a random function – given pairs $(x_i, f(x_1, k)), \ldots, (x_m, f(x_m, k))$, an adversary cannot predict $f(x_{m+1}, k)$ for any x_{m+1}. In other words, F takes a key $k \in K_F$ the set of keys, a n bit sequence $x \in X$ where X is the set $\{0,1\}^n$, and returns a m bit sequence $y \in Y$ where Y is the set $\{0,1\}^m$.

Given the above representation of text document, the search mechanism works as follows. When Alice needs to search for files that contain a word w, she transmits $T(w) = E_{k_p}(w)$ and the key k_c to the server. The server (Bob) searches for w in the index files associated with documents by checking whether $c_i \oplus T(w)$ is of the form $< s, F_{k_c}(s) >$. Bob returns to Alice all the documents that satisfy this condition which can then be decrypted by Alice.

The scheme described above provides secrecy if the pseudorandom function F, the stream cipher used to generate s_i, and the encryption of the document D are secure (i.e., the value t_i are indistinguishable from truly random bits for any computationally bounded adversary). Essentially, the adversary cannot learn the content from either the ciphertext or query logs.

Improvements in Search Efficiency

The approach described above to search over encrypted text requires $O(n)$ cryptographic operations at the server to test if the document contains a given keyword, where n is the number of keywords in the whole document collection. While such an overhead might be tolerable for small documents and small document collections, the approach is inherently not scalable. Subsequently, *Bloom filters* were used to speed up search by reducing time that was proportional to number of keywords to that proportional to number of documents. For each document, the encrypted Bloom filter is queried to check for the presence or absence of the search term which takes $O(1)$ time. To further avoid a complete scan over encrypted documents, another approach based on encrypting the inverted keyword list has been proposed. In this method, first the user constructs the inverted keyword list of his document collection. Then each list associated with a particular keyword is encrypted as follows: Every reference to a document except the first one is encrypted using a key which is stored in the previous node in the list. The first reference to a document is itself encrypted using a key derived from a pseudorandom function applied on the keyword. The collection of keywords are garbled using a pseudorandom permutation applied on the keyword so that the actual physical address of the nodes in the memory looks random. To conduct a search, the user applies the pseudorandom function and the pseudorandom permutation to the keyword to obtain the key and the memory address of the encrypted list of documents associated with that keyword. Then the list is

decrypted node by node. This approach reduces the search time from $O(n)$ to $O(d)$, where n is the total number of documents and d is the total number of documents that actually contain a particular keyword. This approach, however, prevents the user from adding, removing, or updating documents efficiently; since it requires re-encrypting the modified inverted keyword list.

Public-Key Variants

The keyword search over encrypted data problem has been investigated using a public key setup as well. Consider a scenario where emails may consist of sensitive information and hence preferably encrypted by the sender using the recipient's public key. A natural approach would be to instead exploit a public-key encryption technique that directly supports keyword search over encrypted representation. Public key encryption with keyword search (PEKS) has been proposed for the limited context where Alice prespecifies the set of keywords she might be interested in searching the mail based on. The email sender (Bob) can encrypt the email using Alice's public key. Further, PEKS can also be constructed in a black-box way using anonymous identity-based encryption (AIBE). The construction is as follows: (1) The public/private key pair of PEKS is the same as public master secret key of the AIBE scheme. (2) The keywords in PEKS are treated as identities in AIBE. This means that to derive a trapdoor t_w associated with a keyword w, the user has to run the key derivation algorithm of AIBE on input w. (3) To encrypt the keyword w, the user runs the encryption algorithm of AIBE on input 0^k, where k is the security parameter, treating w as the identity who can decrypt the cipher. (4) To search for keyword w, the user decrypts the cipher using t_w. Note that t_w is in fact the secret key associated with w in the AIBE scheme.

Extensions

Another variation to the basic keyword search on encrypted data that has been studied is that of "conjunctive keyword search." Most keyword searches contain more than one keyword in general. The straightforward way to support such queries is to carry out the search using single keywords and then return the intersection of the retrieved documents as the result set. This approach, however, reveals more information than there is a need for and might make the documents more vulnerable to statistical attacks. Therefore, an alternate protocol was developed that returns a document if and only if all the search words are present in it.

There are some shortcomings of all the above methods described for keyword search on encrypted data. Once a capability is given to the untrusted server (or once a search for a certain word has been carried out), that capability can be used forever (by the adversary) to check if these words are present in newly arriving (generated) mails (documents) even though the owner might not want to give the server this capability. This can, in turn, make the schemes vulnerable to a variety of statistical attacks.

Range Queries Over Encrypted Data

Order-Preserving Encryption for 1-d Range Search

For data from an ordered domain like integers, range queries are perhaps the most important class of queries that need to be supported. A one-dimensional range query is issued by specifying the minimum and maximum value of the range, and the server's goal is to return all the data items whose value falls within this range. An order-preserving encryption is one that preserves order amongst elements even after encryption. Given such an encrypted dataset, the range query can be easily mapped to the domain of encrypted values and easily evaluated by the server.

One of the algorithms for order-preserving encryption works in three stages. First, it approximates the distribution of the plaintext data values using a combination of histogram and parametric representations. Then, the data is "flattened" to generate an approximately uniform distribution. Lastly, the flattened values are mapped to a target distribution specified by the data owner which then represents the ciphertext. The third step actually consists of two sub-steps: (1) the target distribution is flattened similar to the first step and scaled to align with the flattened distribution of step 2; (2) the (flattened) values of the plaintext are then mapped to target distribution using the reverse map function generated in sub-step (1). The mapping function is relatively compact and can be represented in space proportional to the number of buckets used to represent the plaintext data distribution. The number of buckets for this is determined using the MDL (Minimum description length) principle, and within each bucket the distribution is approximated by a linear spline (line segment). Intuitively, the number of buckets used is dependent upon the skew of the plaintext value distribution. The "decryption" process is similar but reverse to that of the encryption process.

Yet another way for order-preserving encryption is to generate a random order preserving mapping between two ordered domains, i.e., one corresponding to the plaintext and the other to the ciphertext. The mapping schemes are designed to provide the highest possible security under the given constraint that the mapping should preserve order.

Given a domain interval $\mathcal{D}$ = [1,N] and a range $\mathcal{R}$ = [1,M] where M ≥ N, the encryption algorithm is nothing but a random order-preserving mapping from $\mathcal{D}$ into $\mathcal{R}$. The fact that any such random order-preserving map can be modeled as a negative hypergeometric distributions is used to construct a sampling-based algorithm to generate the encryption function.

Multidimensional Range Queries Over Encrypted Data

The order-preserving schemes described above can be easily extended to support multidimensional range queries in a trivial manner, i.e., by evaluating each dimension separately and taking the intersection of the sets of records retrieved. However, order-preserving encryption itself leaks a lot of statistical information, and when extended to evaluate multidimensional range predicates, it leaks even more information. An alternate scheme to support multidimensional range queries uses a hierarchical tree structure (called *interval tree*). The interval tree is used to choose encryption keys for data items depending upon their key-values and also to determine decryption keys for range predicates. Each predicate evaluation requires scanning the whole table. The leaves of the interval tree correspond to every possible value in the domain of the (ordered) attribute. If the domain size is denoted by T, then the height of the interval tree is $O(\log(T))$. The technique requires storing $O(\log(T))$ encrypted copies for each data item – one associated with each node from the leaf to node path in the interval tree for the associated key value of the data (i.e., on which the search predicate is specified). Similarly, a minimal set of interval tree nodes are used to represent a range $[s,t]$ which is also $O(\log(T))$ in size. This representative set of nodes is denoted by $\wedge(s,t)$. It can be easily shown that the set of nodes on any leaf to root path $\mathcal{P}(x)$ for a value x intersects at exactly one node with the set of nodes in $\wedge(s,t)$ for any range $[s,t]$ containing x. Then, for retrieving all values in range $[s,t]$, the client needs to release all keys associated with nodes in $\wedge(s,t)$. Using these keys, exactly one copy of every value belonging to range $[s,t]$ present in the encrypted dataset can be decrypted and returned to the client. If there are N records in the database, there are $O(N \times \log(T))$ ciphertexts, i.e., $O(\log(T))$ ciphertexts per record.

Information Hiding–Based Approaches to Multidimensional Range Queries

Approaches to support range queries based on exploiting disclosure control mechanisms have also been developed specially in the context of database SQL queries. Disclosure control mechanisms developed in the context of statistical inference control includes techniques such as:

1. *Perturbation*: For a numeric attribute of a record, a random value (chosen from some distribution, like normal with mean 0 and standard deviation σ) is added to the true value.
2. *Generalization*: A numeric or categorical value is replaced by a more general value. For numeric values, it could be an interval that covers the original value and for categorical data, this may be a more generic class, e.g., an ancestor node in a taxonomy tree.
3. *Swapping*: Two different records are selected, and the value of the attribute is swapped (say, the salary value could be swapped between the records corresponding to two individuals).

Information hiding–based approaches to support range queries can potentially use one of these above-mentioned techniques to create additional auxiliary information which is stored alongside encrypted records to facilitate the evaluation of comparison operators on the server.

Techniques based on information hiding mechanisms differ from the cryptographic schemes discussed above in two important ways. First, the nature of disclosure is different in the two approaches. In the latter, the disclosure risk is inversely proportional to the difficulty of breaking the encryption scheme, and if broken, there is complete disclosure of the plaintext values. In contrast, the information disclosure for information hiding approaches could be partial or probabilistic in nature. That is, there could be a non-negligible probability of learning a close approximation to the true value given the transformed data, e.g., the bucket identity in the context of generalization-based approach might disclose a range of values encompassing the true value of the sensitive attribute. Furthermore, the design goals of the two approaches are also different. While cryptographic techniques aim to prevent any leakage of confidential data from the encrypted representation at the server, information hiding–based approaches empower the user to trade-off degree of leakage with overhead of the scheme. The key feature of such techniques is that one can control the disclosure of sensitive information against the cost of executing the queries by selecting appropriate parameter values. The higher the cost a client is willing to pay for evaluating a query, the less information needs to be disclosed to the server in general. Such a "sliding scale" confidentiality makes such schemes particularly attractive to integrate within standard database technology wherein range queries are part of a larger SQL queries involving selections, joins, projections, etc.

Bucketization for Relational Queries

The most popular of the data-hiding techniques is *data generalization*. The resulting view of the data is often referred to as the *bucketized* representation of the data. The bucketization of an attribute corresponds to partitioning its domain into a set of buckets. The strategy used to split the domain into a set of buckets has implications on both the efficiency of the resulting query processing as well as on the disclosure risk of sensitive information to the server. For instance, given a sensitive numeric attribute like *salary*, which has been bucketized, if the adversary somehow comes to know the maximum and minimum values occurring in the bucket, then partial disclosure of sensitive values has already occurred since he can ascertain the range to which the values in the bucket belong to. Further, if he has knowledge of distribution (frequencies) of values in the bucket, he may also be able to ascertain the value of specific records with different degrees of confidence. A natural question is how much information does the generalized representation of data reveal? This depends upon the granularity at which data is partitioned. For instance, assigning all values in the domain to a single bucket will make the bucket-label completely non-informative. However, such a strategy will require the client to retrieve every record from the server. On the other extreme, if each possible data value has a corresponding bucket, the client will get no confidentiality although the records returned by the server will contain no false positives. There is a natural trade-off between the performance overhead and the degree of disclosure.

Privacy-Aware Bucketization

Privacy-aware bucketization is basically a data partitioning technique to play the trade-off between disclosure-risk and performance overhead in query processing. Typically, bucketization develops a strategy that minimizes the disclosure-risk while bounding the performance degradation within a small factor of the optimal cost. Notice the dual of the problem, i.e., where one maximizes the performance with a constraint on information disclosure, could also be addressed once one agrees on the metric for information disclosure. However, such an articulation of the problem has not been studied in the literature. Specifically, *entropy* and *variance* of the value distributions in the bucket are used as measures of disclosure risk (rather the inverse of it, i.e., higher the entropy and variance, lower the disclosure-risk). Intuitively, entropy captures the notion of uncertainty associated with a random element chosen with a probability that follows a certain distribution. The higher the value of entropy of a distribution, the greater is the uncertainty regarding the true value of the element. Further, for an ordered domain, entropy alone does not capture the notion of distance between two values. That is, greater the spread of each bucket distribution, better is the protection against disclosure. Therefore, *variance* of the bucket distribution is also used as an independent measure of disclosure-risk associated with each bucket. Bucketization algorithms have been developed that take the number of buckets as the input parameter and partition the data such that the value of entropy and variance of the bucket-level value distributions are maximized while performance degradation remains within specified bounds. Performance cost is typically measured by the number of false positives returned to the client for a query. The number of buckets is the critical factor that decides what trade-off can be achieved between privacy and performance.

Open Problems

While much progress in research has been made over the past few years on search over encrypted representation, many further challenges remain, like techniques to support dynamic updates and pattern matching queries over text data. A detailed security analysis including the nature of attacks as well as security/confidentiality guarantees supported by different schemes needs to be studied.

Finally, this entry focused on techniques to implement queries over encrypted representation of text documents and relational data. While techniques described prevent (and/or limit) direct leakage of information from the way data is stored or the way the queries are processed, some inference channels still remain. for instance, the server can monitor the queries made by the clients and perform different traffic analysis tasks. For example, if a sequence of queries is always followed by a stock-exchange action, a curious server can learn about the content of these queries, even though they are encrypted, and predict the user action when the same (or similar) sequence of queries appears again. Moreover, it is possible to analyze the importance of different areas in the database, e.g., by counting the frequency of the client accessing the same data items. In order to protect against these types of attacks, one must hide the access patterns of users of the storage system. This problem is related to the classic result on oblivious simulation of Turing machines. In the context of RAM, the goal is to hide the access pattern of a program to memory in order to prevent reverse engineering of the software. Using oblivious RAM techniques for search on encrypted data, one can achieve the highest security guaranties that even hide the access pattern of the clients. However the best-known results on oblivious RAM requires twice the size of the encrypted data to store "fake" data on server side and $O(\log^2(n))$ extra computation per query access.

Recommended Reading

1. Chor B, Goldreich O, Kushilevitz E, Sudan M (1995) Private information retrieval. In: Proceedings of IEEE symposium on foundations of computer science
2. Song D, Wagner D, Perrig A (2000) Search on encrypted data. In: Proceedings of IEEE SRSP
3. Hore B, Mehrotra S, Tsudik G (2004) A privacy-preserving index for range queries. In: International conference on very large databases (VLDB), Toronto
4. Curtmola R, Garay JA, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM conference on computer and communications security, Alexandria
5. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: STOC, Betheseda
6. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious RAMs. J ACM 43(3):431–473
7. Shen E, Shi E, Waters B (2009) Predicate privacy in encryption systems. In TCC, San Francisco
8. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2008) Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J Cryptol 21(3): 350–391
9. Boldyreva A, Chenette N, Lee Y, O'Neill A (2009) Order-preserving symmetric encryption. In: EUROCRYPT

Second Preimage Resistance

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Weak collision resistance

Related Concepts

►Collision Resistance; ►Hash Functions; ►One-Way Function; ►Preimage Resistance; ►Universal One-Way Hash Function

Definition

Second preimage resistance is the property of a ►hash function that it is computationally infeasible to find any second distinct input that has the same output as a given input.

Background

The concept of ►one-way functions in cryptography is attributed to Jevons (1874). Second preimage resistance of hash functions has been introduced by Rabin in 1978 [5].

Theory

Second preimage resistance is related to ►preimage resistance and one-wayness; however, the latter concept is typically used for functions with input and output domain of similar size (►one-way function). A minimal requirement for a hash function to be second preimage resistant is that the length of its result should be at least 90 bits (in 2011). A hash function is said to be a one-way hash function (OWHF) if it is both preimage resistant and second preimage resistant. The relation between ►collision resistance, second preimage resistance, and ►preimage resistance is rather subtle, and it depends on the formalization of the definition: it is shown in [6, 7] that under certain conditions, ►collision resistance implies second preimage resistance and second preimage resistance implies ►preimage resistance.

In order to formalize the definition, one needs to specify according to which distribution the first element in the domain is selected and one needs to express the probability of finding a second preimage for this element. Moreover, one often introduces a class of functions indexed by a public parameter, which is called a ►key. One could then distinguish between three cases: the probability can be taken over the random choice of elements in the range, over the random choice of the parameter, or over both simultaneously. As most practical hash functions have a fixed specification, the first approach is more relevant to applications. The second case is known as a ►Universal One-Way Hash Function or UOWHF.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory IT22(6):644–654
2. Merkle R (1979) Secrecy, authentication, and public key systems. UMI Research, Ann Arbor
3. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral dissertation, Katholieke Universiteit, Leuven
4. Preneel B (1999) The state of cryptographic hash functions. In: Damgard I (ed) Lectures on data security. Lectures notes in computer science, vol 1561. Springer, Berlin, pp 158–182
5. Rabin MO (1978) Digitalized signatures. In: Lipton R, DeMillo R (eds) Foundations of secure computation. Academic, New York, pp 155–166
6. Rogaway P, Shrimpton (2004) Cryptographic hash function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy BK, Meier W (eds) Fast software encryption. Lecturer notes in computer science, vol 3017. Springer, Berlin, pp 371–388
7. Stinson DR (2006) Some observations on the theory of cryptographic hash functions. Designs Codes Cryptogr 38(2): 259–277

Secondary Use Regulations

Laurent Bussard
European Microsoft Innovation Center, Aachen, Germany

Related Concepts

►Data Handling; ►Platform for Privacy Preferences (P3P); ►Social Perspectives on Information Privacy

Definition

►Access control specifies conditions to fulfill before getting access to data. *Usage control* defines how data are handled when access is granted. In order to protect the ►privacy of users (*data subjects*) providing *personal data* to services (*data controllers*), usage control (or *data handling*) is associated with data. In the context of ►privacy policies and usage control, the term "secondary use" means using personal data in a different way than the primary reason why they were collected.

Background

The notion of secondary use was formalized in P3P [1]. In privacy policy languages and usage control, two types of secondary uses are described:

1. *Secondary purpose*: a party collects personal data for a *primary purpose* and also uses them for a *secondary purpose*. For instance, an address is required for delivery but is also used for statistics on customer geographical distribution.
2. *Secondary disclosure* (or *downstream data sharing*): a party collects personal data and discloses them to a *secondary recipient*. For instance, a credit card number is required for payment and is shared with a fraud detection service.

Theory and Applications

Secondary purpose and secondary disclosure can be used together. However, for the sake of simplicity they are described separately in the remaining of this entry.

Secondary Purpose

Data handling specifies the behavior of data controllers with respect to collected data in terms of obligations (e.g., collected data must be deleted within a given data retention period), and in terms of rights (e.g., collected data may be used for given purposes).

Two types of purposes are defined: *primary purposes* describe why data are required and *secondary purposes* provide additional usage details. For instance, a credit card number is required for "Payment and Transaction Facilitation" (primary purpose) could be then used for "Pseudonymous Analysis" (secondary purpose).

While primary purposes are directly related to the service offered by the data controller, secondary purposes disclose more information on the service logic and dataflow. As a result, secondary purposes are often technical. For instance, secondary purposes in P3P 1.1 are "Web Site and System Administration," "Research and Development," "One-Time Tailoring," "Pseudonymous Analysis," "Pseudonymous Decision," "Individual Analysis," "Individual Decision," "Contacting Visitors for Marketing of Services or Products," "Historical Preservation," "Contacting Visitors for Marketing of Services or Products Via Telephone," and an extension point "Other Uses."

Figure 1 shows an example where a patient provides medical data to a health service, which uses it for primary purpose "health" and for secondary purpose "statistic." In this example both purposes (health and statistics) must be expressed by the policy of the health service.

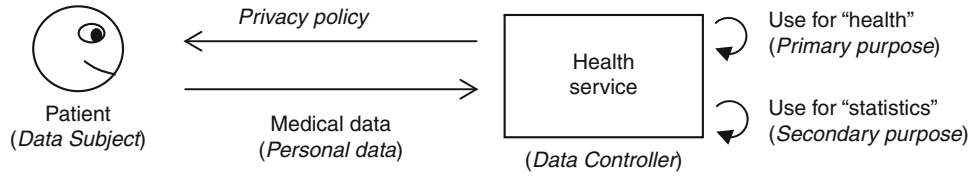
Secondary Disclosure

Data handling specifies whether collected data can be shared with third parties. In this case, personal data subject to data handling is crossing trust domain boundaries. For instance a book store can share collected personal data with the shipping company in charge of delivering orders. Similarly, a health service can share personal data with an insurance to get additional services.

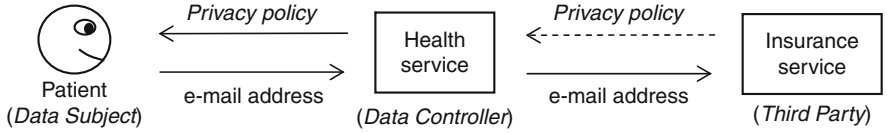
Figure 2 shows an example where a patient provides personal data to a health service, which uses it locally and discloses it to an insurance service. In this example, both data controllers (health and insurance services) must be mentioned as recipient in the privacy policy of the health service.

The fact that collected data may be disclosed to third parties has to be communicated to the user and is thus expressed in the privacy policy of the data controller. P3P 1.1 [1] lets specify classes of third parties to which collected data may be disclosed: "delivery services possibly following different practices," "legal entities following our practices," "legal entities following different practices," "unrelated third parties," and "public fora." There is a trend in ongoing research on privacy policy languages to provide more control on secondary disclosure.

Data handling relying on classes of third parties to specify secondary disclosure is often not precise enough. Recent research works address this problem by extending data handling with access control rules on attributes of third parties [2–5], by taking into account privacy policies



Secondary Use Regulations. Fig. 1 Example of secondary purpose



Secondary Use Regulations. Fig. 2 Example of secondary disclosure

of third parties [3, 4], and/or by specifying chains of secondary disclosure [3].

Recommended Reading

1. W3C Working Group (Nov 2006) The platform for privacy preferences 1.1 (P3P1.1) specification. http://www.w3.org/TR/P3P11
2. Ardagna CA, Cremonini M, De Capitani di Vimercati S, Samarati P (2008) A privacy-aware access control system. J Comput Secur 16(4):369–397
3. Bussard L, Neven G, Preiss FS (Jul 2010) Downstream usage control. In: Proceedings of IEEE policy 2010
4. Becker MY, Malkis A, Bussard L (Dec 2010) A practical generic privacy language. In: Proceedings of sixth international conference on information systems security (ICISS 2010), Gandhinagar
5. Pretschner A, Schuetz F, Schaefer C, Walter T (Jun 2008) Policy evolution in distributed usage control. In: Fourth international workshop on security and trust management. Elsevier, Trondheim

Secret Key Cryptosystem

►Symmetric Cryptosystem

Secret Sharing Schemes

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Synonyms

SSS

Related Concepts

►Access Structure; ►Shamir's Threshold Scheme; ►Threshold Cryptography; ►Visual Secret Sharing Scheme

Definition

A secret sharing scheme makes it possible to share a secret among a group of people in such a way that only well-defined combinations of people can recover the secret.

Background

Informally speaking, a secret sharing scheme (SSS, for short) allows one to share a secret among n participants in a such a way that some sets of participants called *allowed* coalitions can recover the secret exactly, while any other sets of participants (*non-allowed* coalitions) cannot get any additional (i.e., a posteriori) information about the possible value of the secret. The SSS with the last property is called *perfect*. The set Γ of all allowed coalitions is called an ►access structure.

Theory

The history of SSS began in 1979 when this problem was introduced and partially solved by G.R. Blakley [1] and A. Shamir [2] for the case of (n, k)-►threshold schemes, where the access structure consists of all sets of k or more participants. Consider the simplest example of (n, n)-threshold scheme. There is a dealer who wants to distribute a secret s_0 among n participants. Let s_0 be an element of some finite additive ►group G. For instance, G is the group of binary strings of length m with addition by modulo 2, i.e., $G = GF(2)^m$ (►Finite Field). The dealer generates a random sequence $s_1, \ldots, s_n$ such that $\sum_{i=1}^{n} s_i = s_0$ (for instance, by generating independently and uniformly elements $s_1, \ldots, s_{n-1} \in G$ and then putting $s_n := s_0 - \sum_{i=1}^{n-1} s_i$).

Then the dealer sends *privately* to each i-th participant the elements s_i called ►share, i.e., other participants have no information about the value of s_i. It is easy to see that any coalition of less then n participants has no information except of a priori information about s_0 and all participants together recover the value of the secret as $\sum_{i=1}^{n} s_i$. These simple schemes appear to be enough for the realization of arbitrary monotone (i.e., if $A \in \Gamma$ and $A \subset B$, then $B \in \Gamma$) access structure Γ. Namely, see [3], for any allowed coalition $A \in \Gamma$ let the above realize (independently) an $(|A|, |A|)$- threshold scheme, i.e., send to the i-th participant as many shares s_i^A as the number of allowed coalitions to which this participant belongs (it is enough to consider only minimal allowed coalitions).

The probabilistic model of an SSS for the general case is the following (see [4, 5]). There are $n+1$ sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n$ and the probability distribution P on their Cartesian product $\mathcal{S} = \mathcal{S}_0 \times \cdots \times \mathcal{S}_n$. A pair $(P, \mathcal{S})$ is called a perfect probabilistic SSS realizing the access structure Γ if the following two properties hold:

- Participants of an allowed set A (i.e., $A \in \Gamma$) together can recover the secret exactly (formally, $P(S_0 = c_0 \mid S_i = c_i, i \in A) \in \{0, 1\}$ if $A \in \Gamma$).
- Participants forming a non-allowed set A ($A \notin \Gamma$) cannot get additional information beyond their *a priori* information about s_0, i.e., $P(S_0 = c_0 \mid S_i = c_i, i \in A) = P(S_0 = c_o)$ if $A \notin \Gamma$.

These conditions can be reformulated in the language of entropy (►Information Theory) as $H(S_i, i \in A \cup 0) = H(S_i, i \in A) + \delta_\Gamma(A) H(S_0)$, where $\delta_\Gamma(A) = 0$ if $A \in \Gamma$, and $\delta_\Gamma(A) = 1$ otherwise.

There are also combinatorial models of SSSs. An arbitrary set $V \subset \mathcal{S}$ is called the "code" of the combinatorial SSS, and its codewords called "sharing rules." The simplest combinatorial model demands that at first, for every set $A \in \Gamma$, the 0-th coordinate of any codeword from V is uniquely determined by the values of the coordinates from the set A and, secondly, for every set $A \notin \Gamma$ and for any given selection of values of the coordinates from the set A, the number of codewords with given value of 0-th coordinate does not depend on this value. This model is a particular case of the probabilistic model, namely, when all nonzero values of P are equal. The most general definition of combinatorial models, which contains probabilistic ones as a particular case, was given in [6, 7].

For both types of models the "size" of share, provided to any participant, cannot be smaller than the "size" of the secret, where "size" is defined as $log|S_i|$ or $H(S_i)$, respectively, for combinatorial and probabilistic statements of the problem. Special attention has been paid to so-called *ideal* SSSs, where the size of any share coincides with the size of the secret. For example, any (n, k)-threshold scheme can be realized as an ideal perfect SSS (►Threshold Scheme). It was shown in a chain of papers [6–9] that the access structures of ideal perfect SSS correspond to a special class of matroids [10]. On the other hand, any acces structure can be realized as a perfect SSS but probably not very efficient (ideal). At least the above-given realization of [3] demands for some access structures to distribute shares which size is exponentially (in n) larger than the secret's size. An infinite family of access structures was constructed such that for *any* perfect realization, the size of the shares is at least $n/\ln n$ times larger than the size of the secret [11].

To generate shares, the dealer of an SSS has to use some source of randomness, say $r \in X$, where X is in some probabilistic space, and any share s_i is a function of s_0 and r, i.e., $s_i = f_i(s_0, r)$. A linear realization of SSS (or, *linear* SSS) means that all functions $f_i(.)$ are linear. To make it formal: let $s_0, \ldots, s_n$ be elements in m_i-dimensional vector spaces ($i = 0, 1, \ldots, n$) over some ►finite field $GF(q)$ of q elements, and let r be an element of the l-dimensional vector space over the same field. Then a linear SSS is generated by some $(m_0 + l) \times m$ matrix G according to the formula $s = (s_0, \ldots, s_n) = xG$, where $m = \sum_{i=0}^{n} m_i$ and vector x is the concatenation of vectors s_0 and r. Consider vector spaces $V_0, \ldots, V_n$, where V_i is the linear subspace generated by the columns of G that correspond to s_i, i.e., by columns g_j, where $j = m_0 + \cdots + m_{i-1} + 1, \ldots, m_0 + \cdots + m_i$. Then matrix G realizes the access structure Γ perfectly if and only if [12, 13]:

- For any set $A \in \Gamma$, the linear span of subspaces $\{V_a : a \in A\}$ (i.e., the minimal vector subspace containg all these subspaces V_a) contains the subspace V_0.
- For any set $A \notin \Gamma$, the linear span of subspaces $\{V_a : a \in A\}$ intersects with the linear subspace V_0 only by the vector $\mathbf{0}$.

All aforementioned examples of SSS are linear. Note that if all dimensions m_i are equal to 1 then the matrix G can be considered as a generator matrix of a linear error-correcting code (►Cyclic Codes). In particular, it gives another description of ►Shamir's threshold schemes via Reed–Solomon codes [14]. This "coding theory approach" was further developed and generalized to the case of arbitrary linear codes and their minimal words as only possible access structures [16]. Surely, a linear SSS with all dimensions $m_i = 1$ is ideal, but multidimensional linear SSSs with all $m_i = m_0$ give a larger class of ideal SSS [15]. It

is an open question if any ideal SSS can be realized as a (multidimensional) linear ideal SSS.

Modifications of assumptions of the secret sharing problem's statement such as perfectness of the scheme, honesty of the dealer and participants, sending shares via secure, private channels, and so on lead to many variations of secret sharing problem. Among them: *ramp schemes*, SSS with cheaters, publicly verifiable schemes, SSS with public reconstruction [17], and ►visual secret sharing schemes.

Recommended Reading

1. Blakley R (1979) Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 national computer conference, vol 48. New York, pp 313–317
2. Shamir A (1979) How to share a secret. Commun ACM 22(1):612–613
3. Ito M, Saito A, Nishizeki T (1993) Multiple assignment scheme for sharing secret. J Cryptol 6:15–20
4. Karnin ED, Greene JW, Hellman ME (1983) On secret sharing systems. IEEE Trans Inform Theory 29(1):231–241
5. Capocelli RM, De Santis A, Gargano L, Vaccaro U (1993) On the size of shares for secret sharing schemes. J Cryptol 6: 157–167
6. Brickell EF, Davenport DM (1991) On the classification of ideal secret sharing schemes. J Cryptol 4:123–134
7. Blakley GR, Kabatianski GA (1997) Generalized ideal secret sharing schemes and matroids. Prob Inform Transm 33(3): 102–110
8. Kurosawa K, Okada K, Sakano K, Ogata W, Tsujii S (1993) Nonperfect secret sharing schemes and matroids. In: Advances in cryptology EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 126–141
9. Jackson W-A, Martin KM (1998) Combinatorial models for perfect secret sharing schemes. J Combin Math Combin Comput 28:249–265
10. Welsh DJA (1976) Matroid theory. Academic Press, London
11. Csirmaz L (1994) The size of a share must be large. In: Advances in cryptology EUROCRYPT'94., Lecture notes in computer science, vol 950. Springer, Berlin, pp 13–22
12. Blakley GR, Kabatianski GA (1994) Linear algebra approach to secret sharing schemes. In: Error control, cryptology and speech compression. Lecture notes in computer science, vol 829. Springer, Berlin, pp 33–40
13. van Dijk M (1995) A linear construction of perfect secret sharing schemes. In: Advances in cryptology EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 23–34
14. McEliece RJ, Sarwate DV (1981) On secret sharing and Reed-Solomon code. Commun ACM 24:583–584
15. Ashihmin A, Simonis J (1998) Almost affine codes. Design Code Cryptogr 14(2):179–197
16. Massey J (1993) Minilal codewords and secret sharing. In: Proceedings sixth joint Swedish-Russian workshop on information theory. Molle, Sweden, pp 246–249
17. Beimel A, Chor B (1998) Secret sharing with public reconstruction. IEEE Trans Inform Theory 44(5):1887–1896

Secure Audit Logs

Gerardo Pelosi
Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano, Milano, Italy
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici (DIIMM), University of Bergamo, Dalmine (BG), Italy

Synonyms

Secure logging

Related Concepts

►Forward-Security; ►Keyed Hash Function; ►One-Way Function; ►Public Key Cryptography; ►Sequential Signature Aggregation

Definition

An audit log is a chronological record of the activities occurring within an information system. A secure audit log prevents the modification of log data by an unauthorized entity, providing integrity checks over the log content.

Background

An audit log is constituted of a sequence of log entries, each of which is related to a noteworthy event that has occurred within the monitored system. Collecting log data, either at regular intervals or on an event-driven basis is fundamental to reconstruct valuable snapshots of past and current states of the system in order to analyze both regular and anomalous behaviors, allow response to erroneous behaviors, and ensure compliance with established policies and procedures.

Therefore, as relevant part of any IT system, the audit log represents a prime target to effectively break through security barriers arranged to protect sensitive information.

A cryptographic solution for secure logging allows the state of an untrusted system to be audited in order to detect unauthorized past activities by either insider or outsider attackers. Nevertheless, a cryptographic scheme cannot prevent attacks. Preemptive countermeasures must necessarily rely upon the availability of tamper-resistant hardware and/or secure communication channels. Tamper-resistant hardware has been employed in securing database systems maintained in untrusted environments [7], and in storage-based intrusion detection systems (IDS) [8], where the IDS component embedded in the storage device's firmware remains active even if an external source executes software with the same privileges of the operating system. Other traditional methods

to protect logs from tampering include write once read many (WORM) optical drives, and continuous-feed printers; whereas, replication of logged data on many remote trusted servers forces an attacker to break into all or most of the remote hosts in order to erase evidence of her actions [2].

Theory

The detection of both unauthorized modification and intentional destruction of log entries implies the use of integrity mechanisms which need to be irrevocably tamper-evident. Besides typical data integrity requirements, a secure log ought to satisfy a log stream integrity property in order to ensure that the log entries are in the same chronological order as recorded by the system component responsible for logging.

In practical scenarios, audit logs are generated and stored on an untrusted host U (which is assumed to be not sufficiently tamper-resistant); later in time, the audit data are relayed to a trusted server T, who is in charge of validating the received audit log. A moderately trusted entity V (authorized by T through a verification key) may need to verify the authenticity of the audit data while they are still on U, in case there is no high bandwidth constant throughput channel between U and T.

Whenever an attacker gains access to the untrusted logging machine U, she may gain the necessary privileges to modify either archived data or audit log entries without any restrictions, before the communications with T takes place. Thus, in order to allow either V or T to detect any integrity violation, a secure version of the audit log, stored on U, requires a forward-secure stream integrity mechanism to be employed [1, 3, 6]. Forward-secure stream integrity refers to the incapability of the attacker to modify, delete, or reorder the existing log entries, as well as create new bogus entries with logging time preceding her break-in time without being detected.

The forward-secure sequential aggregate (FssAgg) authentication scheme defined by Ma and Tsudik [6] provides an effective secure logging approach, which is (1) forward secure, (2) does not rely on trusted servers to aid V in verifying log integrity, and (3) defends against the deletion of a contiguous subset of most recent log entries generated right before the attacker intrusion.

The FssAgg authentication scheme improves upon previous approaches [1, 3] and represents the most complete solution for audit log authentication.

In the FssAgg authentication scheme, a sequence of signatures (created through a keyed hash function or a digital signature function) is sequentially combined into a single aggregate signature. Only the last aggregate signature is kept in the audit log, whereas previous aggregate signatures are securely deleted.

A successful validation of an aggregate signature is equivalent to the validation of each signature along the chain used to generate it; whereas, an unsuccessful validation implies that at least one component signature is invalid and therefore at least one log entry is not verified.

U is assumed not to be compromised, at initialization time, when an audit log is opened. U uses a key generation algorithm to generate the initial cryptographic keys and transmits them to T over a secure channel; thereafter, U records the first log entry as a fixed message and computes the corresponding signature using the initial signing key. The current signing key is then updated applying a one-way function to it. In this way V or T can detect the potential total deletion of the audit log by an attacker.

At each occurrence of a new log entry, U employs an aggregate signing algorithm that given a message, the current aggregate signature over the previous log entries and a signing key, computes the aggregate signature on all entries and erases the old one. Then, the key-update algorithm uses the one-way function to compute the new forward-secure key, and safely deletes the old one.

The aggregate signing algorithm provides forward-secure stream integrity because an attacker cannot forge the aggregate signature without knowing any signing key employed prior to her break-in. Therefore, the computation of a single aggregate signature for all entries of the audit log makes the alteration or deletion of any log entry, readily detectable by V or T.

Finally, in order to detect attacks aiming at incapacitating the logging system, U creates a closing log entry to inform auditors about the service continuity of the logging process. V and T execute a verification algorithm that, given a verification key and a secure audit log, computes a binary value indicating whether the audit log aggregate signature is valid or not.

The FssAgg authentication scheme can be enforced adopting either a symmetric-key [4, 5] or a public-key paradigm [5, 6]. However, when using a symmetric-key solution, a malicious verifier V may tamper with the log without being detected by other verifiers. Such a tampering can be detected with the help of T, even if much later in time. This drawback is overcome adopting a public-key variant of the FssAgg authentication scheme [5, 6].

Recommended Reading

1. Bellare M, Yee B (2003) Forward-security in private-key cryptography. In: Joye M (ed) Proceedings of the 2003 RSA conference on The cryptographers' track (CT-RSA'03). Lecture Notes in

Computer Science, vol 2612. Springer-Verlag Berlin Heidelberg, New York, NY, USA, pp 1–18
2. Herlihy M, Tygar JD (1988) How to make replicated data secure. In: Pomerance C (ed) A conference on the theory and applications of cryptographic techniques on advances in cryptology (August 16–20, 1987). Lecture notes in computer science, vol 293. Springer-Verlag, London, pp 379–391
3. Kelsey J, Schneier B (1999) Minimizing bandwidth for remote access to cryptographically protected audit logs. In: Proceedings of the recent advances in intrusion detection (RAID'99)
4. Ma D, Tsudik G (2007) Extended abstract: Forward-secure sequential aggregate authentication. In: Proceedings of the 2007 IEEE symposium on security and privacy (SP'07). IEEE Computer Society, Washington, DC, USA, pp 86–91
5. Ma D (2008) Practical forward secure sequential aggregate signatures. In: Proceedings of the 2008 ACM symposium on Information, computer and communications security (ASIACCS '08). ACM, New York, NY, USA, pp 341–352
6. Ma D, Tsudik G (2009) A new approach to secure logging. Trans. Storage 5, 1, Article 2 (March 2009). ACM, New York, NY, USA, 21 p
7. Maheshwari U, Vingralek R, Shapiro W (2000) How to build a trusted database system on untrusted storage. In: Proceedings of the 4th conference on symposium on operating system design and implementation - volume 4 (OSDI'00), vol 4. USENIX Association, Berkeley, CA, USA, pp 10–11
8. Pennington AG, Strunk JD, Griffin JL, Soules CAN, Goodson GR, Ganger GR (2003) Storage-based intrusion detection: watching storage activity for suspicious behavior. In: Proceedings of the 12th conference on USENIX security symposium - Volume 12 (SSYM'03), vol 12. USENIX Association, Berkeley, CA, USA, pp 10–11

Secure Code Dissemination in Wireless Sensor Networks

Sangwon Hyun, Peng Ning
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Synonyms

Secure remote programming

Related Concepts

►Broadcast Authentication; ►Denial-of-Service (DoS) Attack; ►Hash Chain; ►Wireless Sensor Network

Definition

Secure code dissemination is the process of securely propagating a new code image to all sensor nodes using the ad hoc wireless network formed by these nodes, while protecting against various malicious attacks such as injection of fake code image and interfering with normal dissemination.

Background

A wireless sensor network consists of a potentially large number of low-cost, low-power, and multifunctional sensor nodes that communicate over short distances through wireless links. It is desirable and sometimes necessary to reprogram sensor nodes through wireless links after they are deployed, due to, for example, the need of removing bugs and adding new functionalities. Such a process is referred to as *code dissemination*.

In hostile environments, code dissemination may face the following threats. Firstly, the adversary may attempt to modify or replace the real code image being propagated to sensor nodes, introducing malicious code into the sensor network. Secondly, for efficiency in code dissemination every sensor node communicates control messages with its neighbors, and the adversary may transmit fake control messages to disrupt efficient code dissemination. Thirdly, if digital signature is used for authenticating a code image, the adversary can inject bogus signature packets into the network, force the nodes that receive such packets to perform expensive signature verifications, and eventually exhaust their limited battery power. Finally, the adversary can launch DoS attacks exploiting possible authentication delays.

Theory

Several approaches [1, 2, 4, 5] have been developed to cope with the threats mentioned above. All of them assume *Deluge* [3], a code dissemination system in *TinyOS* (an open-source operating system for wireless sensor networks), as the underlying code dissemination protocol. In Deluge, a code image is divided into fixed-size pages, and each page is further split into same-size packets. Deluge employs a page-by-page dissemination strategy, in which each sensor node can request a page only after completely receiving every packet in the previous page. This section describes how the existing secure code dissemination schemes address the threats against Deluge.

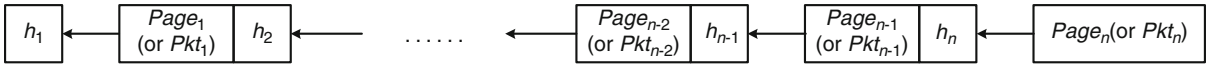
To prevent an adversary to introduce malicious code into the network, all the existing secure code dissemination schemes use cryptographic hash function and digital signature to authenticate a new code image. The base station, which is the starting point of dissemination of a new code image, generates a digital signature to bootstrap the authentication of the code image using its private key. All the existing schemes assume that the adversary is unable to find out the private key of the base station, so it is computationally infeasible for the adversary to forge the base station's signature for his/her own malicious code image. The following are the details of how each scheme authenticates a code image.

In Sluice [5], authentication is done in page granularity. Figure 1 illustrates how Sluice authenticates a code image. Initially, the hash image of the last page($Page_n$) is computed and appended to $Page_{n-1}$. Then the hash image of $Page_{n-1}$ including h_n is computed and appended to $Page_{n-2}$. Sluice repeats such procedure until getting h_1. Finally, h_1 is signed and included in the signature packet along with the digital signature. When receiving a code image, the sensor node first requests the signature packet and verifies h_1 in it through signature verification. Then it requests $Page_1$ and verifies it by comparing the hash image computed from the received $Page_1$ with h_1 already verified. In such way, the node can verify all n pages. In Sluice, each page can be verified only after receiving and verifying the signature packet and all the previous pages. Because of the authentication in a page unit, an individual code dissemination packet cannot be immediately verified upon receipt.

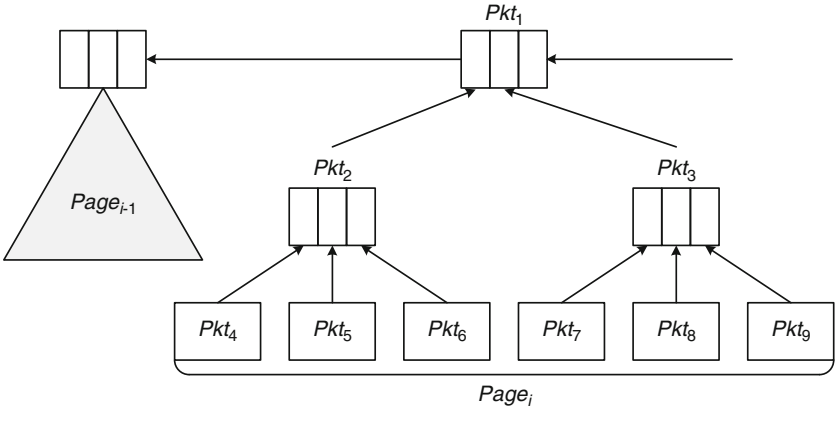
The approach of [2] is similar to Sluice except that the granularity of the authentication is packet rather than page. The hash image of each code dissemination packet is included in the previous packet as illustrated in Fig. 1, and the hash image of the first packet is signed. Therefore, in this approach each packet can be verified only after receiving and verifying all the previous packets including the signature packet. Due to the possibility of out-of-order delivery of the packets in a same page, this approach does not guarantee immediate authentication of each code dissemination packet.

The approach of [1] constructs a Merkle hash tree [6] to authenticate the packets in each page. Figure 2 shows a Merkle hash tree when each page consists of nine packets and each packet payload can include up to three hash images. Each node of the tree corresponds to a single packet. Each code dissemination packet (Pkt_4 to Pkt_9) in a page becomes a leaf node of the tree. An internal node contains the hash images of all its child nodes, and the number of the children of each internal node depends on the number of hash images the corresponding packet payload can include. The hash image of each root packet (Pkt_1) is included in the root packet of the hash tree for the previous page, and the hash image of the root packet of the first page is signed. In order to verify a packet, each node must have received and verified the signature packet and all the packets in the previous pages. In addition, the node must have received and verified all the packets on the path from the root to the parent of the packet in the tree, for instance, Pkt_1 and Pkt_2 to verify Pkt_5. Thus, under the page-by-page dissemination of Deluge, this approach does not guarantee immediate authentication of each packet.

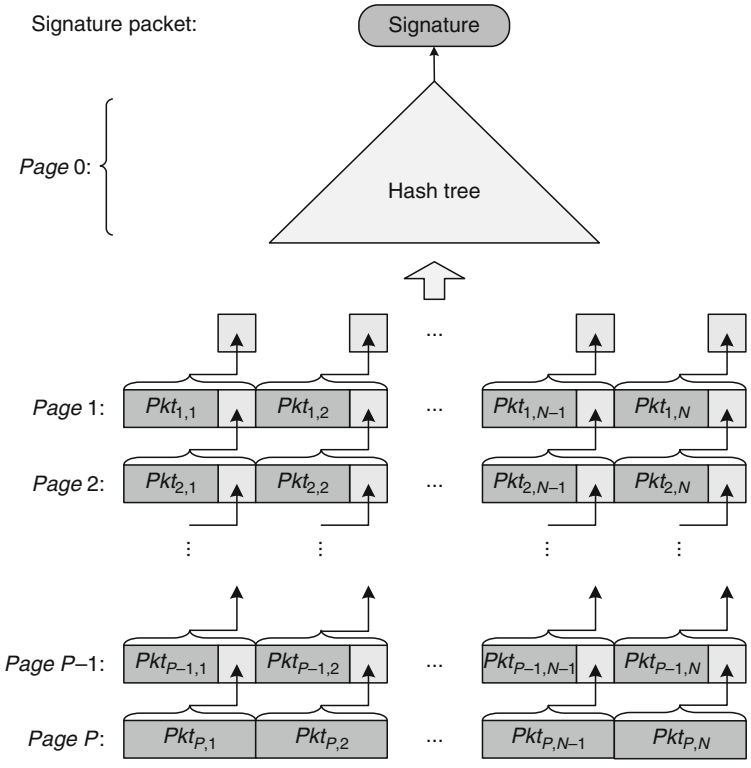
Finally, in Seluge [4], the hash image of each code dissemination packet is included in the corresponding packet in the previous page; in Fig. 3, for instance, the hash image of $Pkt_{2,1}$ in $Page_2$ is included in $Pkt_{1,1}$ in $Page_1$. Seluge also uses a Merkle hash tree as in [1], but relying on a different strategy to authenticate the hash images of the packets in $Page_1$. The packets related to this Merkle hash tree compose $Page_0$. Figure 4 illustrates how Seluge



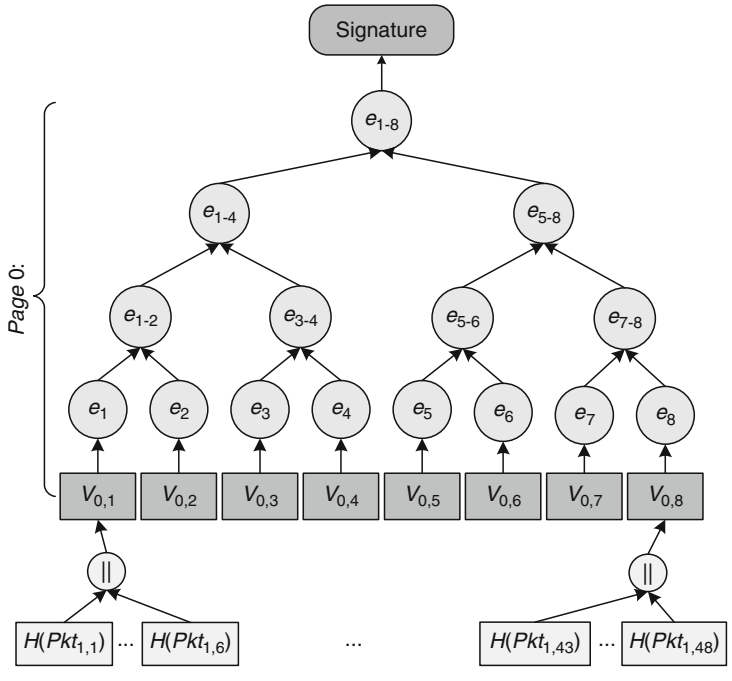
Secure Code Dissemination in Wireless Sensor Networks. Fig. 1 The main concept for authentication in [2, 5]



Secure Code Dissemination in Wireless Sensor Networks. Fig. 2 An example Merkle hash tree of the approach in [1]



Secure Code Dissemination in Wireless Sensor Networks. Fig. 3 Authentication of code images in Seluge



Secure Code Dissemination in Wireless Sensor Networks. Fig. 4 The Merkle hash tree in Seluge

constructs a Merkle hash tree. Initially, the hash images of all the $Page_1$ packets are grouped into chunks (e.g., $V_{0,1} = H(Pkt_{1,1})||\cdots||H(Pkt_{1,6})$), and the hash image of each chunk is computed and associated with a leaf of the tree ($e_1 = H(V_{0,1})$). Each internal node has two child nodes, and its value is the hash image of them ($e_{1-2} = H(e_1||e_2)$, and $e_{1-4} = H(e_{1-2}||e_{3-4})$). Each $Page_0$ packet $Pkt_{0,i}$ from the hash tree consists of $V_{0,i}$ and the values in its authentication path (i.e., the siblings of the nodes in the path from $V_{0,i}$ to the root) in the tree; for example, $Pkt_{0,1}$ consists of $V_{0,1}$, e_2, e_{3-4}, and e_{5-8}. If receiving each $Page_0$ packet, the sensor node can compute the root of the tree from the information included in the packet. The root of the tree is signed and included in the signature packet. In Seluge, after receiving and verifying every packet in a page, all the packets of the next page can be immediately verified upon receipt. As a result, along with the page-by-page dissemination strategy, Seluge provides immediate authentication of every code dissemination packet.

For all the remaining threats, only Seluge includes protection mechanisms. To deal with the attack using fake control messages, Seluge provides local broadcast authentication for control messages using cluster key. To protect from DoS attacks using fake signatures, Seluge includes another layer of protection for signature packet using a weak authentication mechanism called message-specific puzzles [7].

Open Problems

Insider attackers who compromise some regular nodes in a network can still initiate DoS attacks using the information available on the compromised nodes. The current secure code dissemination schemes have no way to verify behavioral integrity of each node. Thus, if received information passes an integrity check, the receiver just trusts it. As a result, the insider attacker can introduce legitimate but fake information into the network using the key information in the compromised nodes.

No secure code dissemination schemes have considered jamming attacks. In addition, they assume that all the nodes in a network use a single channel for at least the control messages. Therefore, by jamming that channel over a certain region of the network, the adversary can disable the propagation of a code image into or significantly slow down the propagation speed over the region. In addition, sophisticated jamming strategies make it hard to detect and so increase the impact of jamming.

Recommended Reading

1. Deng J, Han R, Mishra S (2006) Secure code distribution in dynamically programmable wireless sensor networks. In: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06), April 2006, Nashville, Jennessee
2. Dutta PK, Hui JW, Chu DC, Culler DE (2006) Securing the deluge network programming system. In: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06), April 2006, Nashville, Jennessee
3. Hui JW, Culler D (2004) The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04), November 2004, Baltimore, Maryland
4. Hyun S, Ning P, Liu A, Du W (2008) Seluge: secure and dos-resistant code dissemination in wireless sensor networks. In: Proceedings of the Seventh International Conference on Information Processing in Sensor Networks (IPSN '08), April 2008, St. Louis, Missouri
5. Lanigan PE, Gandhi R, Narasimhan P (2006) Sluice: secure dissemination of code updates in sensor networks. In: Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS '06), July 2006, Cambridge, Massachusetts
6. R. Merkle. Protocols for public key cryptosystems. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, April 1980, Oakland, California
7. Ning P, Liu A, Du W (2008) Mitigating DoS attacks against broadcast authentication in wireless sensor networks. ACM Trans Sensor Networks 4(1):1–35

Secure Communication

►Secure Routing in Wireless Mesh Networks

Secure Computation

►Multiparty Computation

Secure Computer System Model

►Bell–La Padula Model

Secure Coprocessor

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Synonyms

Hardware security module

Related Concepts

►Trusted Computing; ►Zeroization

Definition

The term *secure coprocessor* generally refers to a physically secure subsystem of a larger host computer. The aspiration is that the coprocessor provides a sanctuary for computation and data storage secure even against an adversary who physically controls the host.

The term is usually used without a hyphen.

Background

The concept of "secure coprocessor" emerged from a number of different strands, including financial cryptography, military, and early rights management. The burden of cryptographic operations on a host CPU motivates the use of a separate *crypto coprocessor*; the sensitive nature of keys–coupled with the use of such devices in the financial sector, long used to armor–introduced the idea of physical security. Military computing also brings up the issue of tamper-resistant computing (e.g., consider a sensitive battlefield device that may fall into enemy hands). In the commercial world (in contrast to combat), vendors of "soft" digital objects, such as software, long considered ways to restrict unauthorized use via specialized hardware *dongles*: the idea is that software of interest runs primarily on an open host machine, but it selectively interacts with a primitive coprocessor–whose absence prevents illegitimate. The security of these protections depends on the security of this dongle–and of how the host software interacts with it.

Theory and Applications

Since the goal of a secure coprocessor is to provide protection even when the adversary controls the larger computer, discussions of design and implementation often begin with discussion of physical security techniques, which usually center on ensuring some core secrets are *zeroized* upon attack. Since another goal of a secure coprocessor is house computation securely, other aspects of design wrestle with how to configure and control this computation–and the computation executing on an untampered device can establish that fact to a remote relying party.

For exploring the application space of a secure coprocessor, we might start with the basic taxonomy of considering integrity and/or confidentiality, of data and/or computation. For example, such a device might keep cryptographic keys secret (confidential data), but it might also house a proprietary pricing algorithm (confidential computation) or provide a trustworthy neutral ground to carry out an auction (integrity of computation). Considering the relation of the coprocessor to its host provides another angle to consider things. For example, we might carefully design *partitioned computation* between what executes on the untrusted host and on the trusted coprocessor, with the aspiration that the design of the partitioned coupled with the protections of the coprocessor endows the overall system with some security properties it might otherwise not achieve. We might also try to give the coprocessor some ways to control the host (such as via *busmastering* privileges), so that the coprocessor can assure (or at least measure) the integrity of the rest of the system.

Historically, research at IBM Watson (e.g., [2, 7, 8]) developed the early ideas of secure coprocessing. Work at CMU (e.g., [6, 9]) developed applications; [4] provides additional ideas. The work of Bill Arbaugh (e.g., [1, 3]) explored ways of protecting boot and host integrity. [5] Smith provides a longer history of this technology, culminating in the design, implementation, and validation of the IBM 4758 secure coprocessor platform.

With the emergence of the Trusted Computing Group (and the concomitant ubiquity of TPMS), and the emergence of *late-launch* CPU architectures, many of the early ideas of secure coprocessing are now seeing a revival in mainstream computing.

Recommended Reading

1. Arbaugh W, Farber D, Smith J (1997) A secure and reliable bootstrap architecture. In: Proceedings of the 1997 symposium on security and privacy conference, Oakland, pp 65–71
2. Palmer E (1992) An Introduction to Citadel—A Secure crypto coprocessor for workstations. RC18373, IBM T.J. Watson Research Center 1992
3. Petroni N, Fraser T, Molina J, Arbaugh WA (2004) Copilot—a Coprocessor-based Kernel Runtime Integrity Monitor. In: Proceedings of the 13th USENIX security symposium, USENIX Press, San Diego, pp 179–194
4. Smith SW (1996) Secure coprocessing applications and research issues. Los Alamos Unclassified Release LAUR-96-2805, Los Alamos National Laboratory, August 1996
5. Smith SW (2004) Trusted computing platforms: design and applications. Springer, New York
6. Tygar JD, Yee BS (1991) Strongbox: a system for self-securing programs. In: Rashid RF (ed) CMU computer science: A 25th anniversary commemorative. Addison-Wesley, Reading, pp 163–197
7. White SR, Comerford LD (1987) ABYSS: a trusted architecture for software protection. IEEE Symposium on Security and Privacy, Oakland
8. White S, Weingart SH, Arnold W, Palmer ER (1991) Introduction to the Citadel architecture: security in physically exposed environments. RC16672, IBM T.J. Watson Research Center
9. Yee BS (1994) Using secure coprocessors. Ph.D.thesis, Computer Science, Carnegie Mellon University, May 1994

Secure Data Aggregation

Wensheng Zhang
Department of Computer Science, College of Liberal Arts and Sciences, Iowa State University, Ames, IA, USA

Related Concepts

►Data Aggregation; ►Homomorphic Encryption

Definition

Securing data aggregation is to protect data aggregation, one of basic primitives of sensor networks for communication efficiency, against security attacks such as eavesdropping and forging aggregated results.

Background

In-network data aggregation has been a widely adopted data-centric forwarding mechanism for sensor networks. With this mechanism, each sensor node aggregates the data that is generated by itself and that it is asked to be forwarded, and forwards only the aggregated result. Typical aggregation functions applied here include sum, average, max/min, median, histogram, and so on. By employing in-network data aggregation, the amount of data communicated in the network can be significantly reduced, which consequently decreases bandwidth consumption and energy depletion. Due to the open nature of wireless communication and the often untrusted (even hostile) deployment environment of sensor networks, data aggregation can be attacked by the adversary. Particularly, the adversary may eavesdrop the communication to obtain intermediate aggregation results, and a compromised forwarding node may misbehave in data aggregation.

Theory and Applications

To protect aggregated data from being exposed to any entity other than the sink, i.e., protect the confidentiality/privacy of raw and aggregated data, Castelluccia et al. [1] proposed a simple additively homomorphic encryption technique, and applied it to secure data aggregation in sensor networks. The basic idea is as follows: Each sensor node i shares a unique secret key k_i with the sink, and all nodes share a big number M. When a leaf node i reports a data item m, it applies the additively homomorphic encryption function on the data and sends the encrypted data to its downstream node towards the sink. Specifically, the encrypted data is $m'_i = m + k_i (mod M)$. When a non-leaf node j has received n encrypted reports from its upstream nodes, denoted as m'_{i_t} where $t = 1, \cdots, m$, it computes and reports $m'_j = m_j + \sum_{t=1}^{n} m'_{i_t} + k_j (mod M)$, where m_j is the data item generated by itself. As the sink gets aggregation results from all of its upstream nodes, the sink sums the results up and subtract $\sum_{i \in V} k_i$, where V is the set of all nodes in the network, from the sum to extract the actual aggregation result. This scheme was later enhanced by Zhang et al. [10] for generic privacy-preserving, approximate aggregation. The key idea is that most of numeric data aggregation functions can be approximately implemented as a histogram-based data aggregation plus certain post-aggregation process at the sink. They also develop efficient schemes for privacy-preserving histogram-based aggregation.

As an approach to protecting data confidentiality/privacy in aggregation, He et al. [5] proposed the cluster-based private data aggregation (CPDA) scheme and the slice-mix-aggregate (SMART) scheme. Both schemes rely on the collaboration between neighboring nodes to hide the actual data reported by each individual node. In CPDA, sensor nodes randomly form clusters, and sensor nodes within the same cluster collectively compute the aggregate value. Two rounds of interactions are required: first, each pair of sensor nodes in the same cluster exchange one data item derived from their own sensory data; second, each sensor node broadcasts to other sensor nodes in the same cluster another data item derived from the data it received in the previous round. In the improved SMART scheme, each sensor node only needs to exchange data for once with some nearest sensor nodes; that is, each sensor node slices its sensory data into a certain number (say, n) of pieces, and the pieces are then securely distributed to $n-1$ nearest sensor nodes for aggregation. Both schemes can only tolerate the collusion of up to a certain threshold number (i.e., the number of sensors in a cluster minus two for CPDA, and the sum of out-degree and in-degree minus one for SMART) of sensor nodes.

Another security problem that has been addressed by existing research work is the identification of aggregating nodes that forge aggregated data. Yang et al. [9] proposed a secure hop-by-hop data aggregation protocol (SDAP). The design of SDAP applies the principles of divide-and-conquer and commit-and-attest. The key idea is as follows: First, all nodes in the network are partitioned into several logical groups. The grouping is known only by the sink, each group leader only knows it is a group leader, and other nodes do not have any knowledge about the grouping. In implementation, each logical group is a connected portion of the tree containing all nodes. Second, in each group, readings of nodes are eventually aggregated by the group leader, and the aggregated result of each group is sent to the sink securely without further aggregation. Third, after the sink receives the aggregated results from all logical

groups, it uses Grubbs' test, also known as the maximum normalized residual test, to find out outliers. Fourth, the sink collects commitments from a selected set of individual nodes in each group whose aggregated result is considered an outlier, and attestation is performed based on the commitments to decide if there are any aggregating nodes that have forged aggregated data.

Following the aggregate–commit–prove framework described by Przydatek et al. [7], Chan et al. [3] proposed a secure hierarchical in-network aggregation scheme. The scheme runs in three phases. In the first phase, query dissemination, the sink broadcasts a query to the aggregation tree that contains all nodes in the network. In the second phase, aggregation commit, nodes iteratively construct a commitment structure resembling a hash tree. Specifically, each leaf node reports its data to its parent on the aggregation tree. Each internal node performs the aggregation function on all inputs obtained from its children. In addition, it also computes a commitment that is the root of the hash tree in which the leaves are the inputs from its children. Then, the internal node forwards both its aggregated data and its commitment as an input to its parent. The procedure continues unless the sink has aggregated all the data and produced a commitment for the whole tree. In the third phase, result-checking, each node in the network checks if its contribution is included in the aggregation result. Specifically, the sink broadcasts the final commitment to every node, and every node is also provided information from its upstream nodes that is sufficient for it to verify if its own contribution is used in computing the global hash tree whose root is the broadcast final commitment. If a verification fails in one or more node, misbehavior in aggregation is detected.

Open Problems

Security issues such as data confidentiality/privacy protection and aggregation misbehavior detection have been mostly studied separately. Existing confidentiality/privacy protection schemes cannot detect aggregation misbehaviors, and existing aggregation misbehavior detection schemes cannot protect data confidentiality from insiders. Hence, new solutions are needed to address both issues simultaneously. In addition, other security issues regarding data aggregation need to be discovered and then solved.

Recommended Reading

1. Castelluccia C, Mykletun E, Tsudik G (2005) Efficient aggregation of encrypted data in wireless sensor networks. In: Proceedings of Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), pp 109–117
2. Chan H, Perrig A (2008) Efficient security primitives derived from a secure aggregation algorithm. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), pp 521–534
3. Chan H, Perrig A, Song D (2006) Secure hierarchical in-network aggregation in sensor networks. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), pp 278–287
4. Du W, Deng J, Han Y, Varshney P (2003) A witness-based approach for data fusion assurance in wireless sensor networks. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), pp 1435–1439
5. He W, Liu X, Nguyen H, Nahrstedt K, Abdelzaher T (2007) Pda: Privacy-preserving data aggregation in wireless sensor networks. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM), pp 2045–2053
6. Mahimkar A Rappaport T (2004) Securedav: a secure data aggregation and verification protocol for sensor networks. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), pp 2175–2179
7. Przydatek B, Song D, Perrig A (2003) Sia: secure information aggregation in sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys), pp 255–265
8. Roy S, Conti M, Setia S, Jajodia S (2009) Secure median computation in wireless sensor networks. Ad Hoc Networks, 7:1448–1462
9. Yang Y, Wang X, Zhu S, Cao G (2006) Sdap: a secure hop-by-hop data aggregation protocol for sensor networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp 356–367
10. Zhang W, Wang C, Feng T (2008) Gp^2s: generic privacy-preservation solutions for approximate aggregation of sensor data. In: Proceedings of Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pp 179–184

Secure Data Outsourcing: A Brief Overview

Radu Sion
Network Security and Applied Cryptography Lab, Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Synonyms

Cloud computing

Open Problems and Future Directions

To be practical, outsourced data services should allow expressive client queries (e.g., relational joins with arbitrary predicates) without compromising confidentiality. This is a hard problem because decryption keys cannot be

directly provided to potentially untrusted servers. Moreover, if the remote server cannot be fully trusted, protocol correctness becomes essential.

Definition

Today, sensitive data is being managed on remote servers maintained by third party outsourcing vendors. This is because the total cost of data management is 5–10 times higher than the initial acquisition costs [33].

In data outsourcing [1–6, 6–14] clients are naturally reluctant to place sensitive data under the control of a foreign party without strong security assurances of *correctness*, *confidentiality*, and data access *privacy*.

Background

Query Correctness

Informally, a query mechanism is called *correct* if the server is bound to the sequence of update requests performed by the client. Either the server responds correctly to a query or its malicious behavior is immediately detected by the client: In applied settings, *correctness* in data outsourcing can be often decomposed into two protocol properties, namely, *data integrity* and *query completeness*. Data integrity guarantees that outsourced data sets are not tampered with by the server. Completeness ensures that queries are executed against their entire target data sets and that query results are not "truncated" by servers.

Existing work focuses mostly on solutions for simple one-dimensional range queries, and variants thereof. In a publisher–subscriber model, Devanbu et al. deployed Merkle trees to authenticate data published at a third party's site [32], and then explored a general model for authenticating data structures [45, 46]. Hard-to-forge verification objects are provided by publishers to prove the authenticity and provenance of query results.

In [49], mechanisms for efficient integrity and origin authentication for simple selection predicate query results are introduced. Different signature schemes (DSA, RSA, Merkle trees [47] and BGLS [20]) are explored as potential alternatives for data authentication primitives. Mykletun et al. [50] introduce *signature immutability* for aggregate signature schemes – the difficulty of computing new valid aggregated signatures from an existing set. Such a property is defeating a frequent querier that could eventually gather enough signature data to answer other (un-posed) queries. The authors explore the applicability of signature-aggregation schemes for efficient data authentication and integrity of outsourced data. The considered query types are simple selection queries.

Similarly, in [51], digital signature and aggregation and chaining mechanisms are deployed to authenticate simple selection and projection operators. While these are important to consider, nevertheless, their expressiveness is limited. A more comprehensive, query-independent approach is desirable. Moreover, the use of strong cryptography renders this approach less useful. Often simply transferring the data to the client side will be faster.

In [56], *verification objects* VO are deployed to authenticate simple data retrieval in "edge computing" scenarios, where application logic and data is pushed to the edge of the network, with the aim of improving availability and scalability. Lack of trust in edge servers mandates validation for their results – achieved through verification objects.

In [55], Merkle tree and cryptographic hashing constructs are deployed to authenticate the result of simple range queries in a publishing scenario in which data owners delegate the role of satisfying user queries to a third-party un-trusted publisher. Additionally, in [52], virtually identical mechanisms are deployed in database outsourcing scenarios. Devanbu et al. [31] propose an approach for signing XML documents allowing untrusted servers to answer certain types of path and selection queries.

In [57], a novel method for proofs of *actual* query execution in an outsourced database framework for *arbitrary* queries was proposed. The solution prevents a "lazy" or malicious server from incompletely (or not at all) executing queries submitted by clients. It is based on a mechanism of runtime query "proofs" in a challenge-response protocol. For each batch of client queries, the server is "challenged" to provide a *proof of query execution* that offers assurance that the queries were actually executed with completeness, over their entire target data set. This proof is then checked at the client site as a prerequisite to accepting the actual query results as accurate.

Data Confidentiality

Confidentiality constitutes another essential security dimension required in data outsourcing scenarios, especially when considering sensitive information. Potentially untrusted servers should be able to process queries on encrypted data on behalf of clients without compromising confidentiality.

Searching on Encrypted Data

Confidentiality alone can be achieved by encrypting the outsourced content before outsourcing to potentially access-curious servers. Once encrypted, however, it cannot be easily processed by servers. This limits the applicability

of outsourcing, as the type of processing primitives available will be reduced dramatically. It becomes important to provide mechanisms for server-side data processing. Untrusted servers should be able to query encrypted data on behalf of clients with confidentiality.

One of the first processing primitives that has been explored allows clients to search directly in remote encrypted data [15, 17, 19, 21, 26, 30, 34, 36, 59, 65]. In these efforts, clients either linearly process the data using symmetric key encryption mechanisms, or, more often, outsource additional secure (meta)data mostly of size linear in the order of the original data set. This meta-data aids the server in searching through the encrypted data set while revealing as little as possible.

For example, Song et al. [59] propose a scheme for search on encrypted data in a scenario where a mobile, bandwidth restricted user, wishes to store data (e-mail) on an untrusted server. The scheme requires the user to split the data into fix-sized words, encrypt each word separately using a symmetric key protocol, and xor the result with a structure containing a pseudo-random bit string and a mapping of the string under a secret key, using a pseudo-random function. The secret key is made dependent on the encrypted word. The resulting data is stored on the server. The structure enables the detection of keyword matches, without revealing the server the keyword or the contents of the stored data. The paper also discusses the use of an encrypted index, allowing the whole data to be encrypted as a block.

Eu-Jin Goh [34] proposes to associate indexes with the documents stored by the server. More precisely, the index of a document is a Bloom filter [18] containing a codeword for each unique word in the document. The codeword of a word is derived by twice applying a pseudo-random function to the word. The size of document indexes, as documented in the paper, is proportional to the document size. Chang and Mitzenmacher [26] propose a similar approach, where the index associated with documents consists of a string of bits of length equal to the total number of words used (dictionary size). Two solutions are given, one where the dictionary of words can be stored at the client and one where it has to be stored encrypted at the server.

An interesting version of searching on encrypted data is proposed by Boneh et al. [19], where e-mails encrypted by senders with the public key of the intended receiver are stored on untrusted mail servers. The paper presents protocols allowing receivers to search. In the first protocol, a noninteractive searchable encryption scheme is based on a variant of the Diffie-Hellman problem and uses bilinear maps on elliptic curves. The second protocol, using only trapdoor permutations, needs a large number of public/private key pairs. Both protocols require the individual encryption of each word.

Golle et al. [36] extend the above problem to conjunctive keyword searches on encrypted data. They propose two solutions. The first solution requires the server to store capabilities for conjunctive queries, with sizes linear in the total number of documents. The paper claims that a majority of the capabilities can be transferred offline to the server, but this only assumes that the client knows beforehand its future conjunctive queries. The second solution requires much less communication between the client and the server, proportional with the number of keywords in the conjunctive search, but doubles the size of the data stored by the server. A severe limitation of these schemes is the requirement of specifying the exact positions where the search matches have to occur.

Queries on Relational Data

The paradigm of providing a database as a service emerged [38] likely due in no small part to the dramatically increasing availability of fast, cheap networks.

Hacigumus et al. [38] propose a method to execute SQL queries over partly obfuscated outsourced data. The data is divided into secret partitions and queries over the original data can be rewritten in terms of the resulting partition identifiers; the server can then partly perform queries directly. The information leaked to the server is claimed to be 1-out-of-s where s is the partition size. This balances a trade-off between client-side and server-side processing, as a function of the data segment size. At one extreme, privacy is completely compromised (small segment sizes) but client processing is minimal. At the other extreme, a high level of privacy can be attained at the expense of the client processing the queries in their entirety. Moreover, in [39] the authors explore optimal bucket sizes for certain range queries. Similarly, data partitioning is deployed in building "almost"-private indexes on attributes considered sensitive. An untrusted server is then able to execute "obfuscated range queries with minimal information leakage." An associated privacy-utility trade-off for the index is discussed.

One main concern with such mechanisms is the fact that they leak information to the server, at a level corresponding to the granularity of the partitioning function. For example, if such partitioning is used in a range query, to execute rewritten queries at the partition level, the server will be required to precisely know the range of values that each partition contains. Naturally, increasing partition sizes tends to render this knowledge more fuzzy. This, however, requires additional client-side work in pruning the (now) larger results (due to the larger partitions) etc.

Additionally, for more complex queries, particularly joins, due to the large segments, such methods can feature a communication overhead larger than the entire database.

Ge et al. [61] discuss executing aggregation queries with confidentiality on an untrusted server. Unfortunately, due to the use of extremely expensive homomorphisms (Paillier [53, 54]) this scheme leads to impractically large processing times for any reasonable security parameter settings. Current homomorphisms are simply not fast enough to be usable for practical data processing.

Oblivious Transaction Processing

Existing database outsourcing research mostly considered a unified single client model [32, 49]. In *multi-client application scenarios*, however, additional challenges need to be handled. Specifically, outsourcing systems need to address transaction management on encrypted data. This is a challenging endeavor due to the fact that conflict resolution needs to be performed with the aid of the untrusted service provider which should have only minimal or no access to transaction-specific data.

Williams et al. [64] introduce an Oblivious Transaction Processing (OTP) mechanism for outsourcing the durability property of a multi-client transactional database to an untrusted service provider. Specifically, the work enables untrusted service providers to support transaction serialization, backup, and recovery for clients, with full data confidentiality and correctness. Moreover, providers learn nothing about transactions (except their size and timing), thus achieving read and write access pattern privacy. The main insight behind the OTP mechanisms is to deploy the curious transaction server as a communication and serialization conduit for multi-client scenarios.

Data Access Privacy

In existing protocols, even though data sets are stored in encrypted form on the server, the *client query access patterns* leak essential information about the data. A simple attack can correlate known public information with *hot* data items (i.e., with high access rates), effectively compromising their confidential nature. In competing business scenarios, such leaks can be extremely damaging, particularly due to their unpredictable nature. This is why, to protect confidentiality, it is important to also provide assurances of access pattern privacy.

Private Information Retrieval

Private Information Retrieval (PIR) protocols were first proposed as a theoretical primitive for accessing individual items of outsourced data, while preventing servers to learn anything about the client's access patterns [27]. Chor et al. [27] proved that in information theoretic settings in which queries do not reveal any information about the accessed data items, a solution requires $\Omega(n)$ bits of communication. To avoid this overhead, they show that for multiple *non-colluding* databases holding replicated copies of the data, PIR schemes exist that require only sub-linear communication overheads. This multi-server assumption, however, is rarely viable in practice.

In single-server settings, it is known that PIR requires a full transfer of the database [27, 28] for computationally unbounded servers. For bounded adversaries, however, *computational* PIR (cPIR) mechanisms have been proposed [22, 23, 25, 41–44, 60]. In such settings however, it is trivial to establish an $O(n)$ lower bound on server processing, mandating expensive trapdoor operations per bit, to achieve access privacy. This creates a significant privacy - efficiency trade-off between the required server computation cycles and the time to actually transfer the data and perform the query at the client site.

This trade-off is explored in [58] where the authors discuss single-server computational PIR *for the purpose of preserving client access patterns leakage*. They show that deployment of non-trivial single server private information retrieval protocols on real (Turing) hardware is orders of magnitude more time-consuming than trivially transferring the entire database to the client. The deployment of computational PIR in fact *increases* both overall execution time and the probability of *forward* leakage, when the deployed present trapdoors become eventually vulnerable – e.g., today's access patterns will be revealed once factoring of today's values will become possible in the future.

The authors note that these results are beyond existing knowledge of mere "impracticality" under unfavorable assumptions. On real hardware, *no* existing non-trivial single server PIR protocol could have possibly had outperformed the trivial client-to-server transfer of records in the past, and is likely not to do so in the future either. Informally, this is due to the fact that it is more expensive to PIR-process one bit of information than to transfer it over a network.

Oblivious RAM

Oblivious RAM [35] provides access pattern privacy to clients (or software processes) accessing a remote database (or RAM), requiring only logarithmic storage at the client. The amortized communication and computational complexities are $O(\log^3 n)$. Due to a large hidden constant factor, the ORAM authors offer an alternate solution with

computational complexity of $O(\log^4 n)$, that is more efficient for all currently plausible database sizes.

Secure Hardware-Aided PIR

Following the original ORAM work, Asonov introduced [16] a PIR scheme that uses a secure CPU to provide (an apparent) $O(1)$ online communication cost between the client and server. However, this requires the secure CPU on the server side to scan portions of the database on every request, indicating a computational complexity cost of $O(n)$, where n is the size of the database.

An ORAM-based PIR mechanism is introduced by Iliev and Smith [40], who deploy secure hardware to achieve a cost of $O(\sqrt{n}\log n)$. This is better than the poly-logarithmic complexity granted by ORAM for the small database sizes they consider. This work is notable as one of the first full ORAM-based PIR setups.

An improved ORAM-based PIR mechanism with $O(n/k)$ cost is introduced in [62], where n is the database size and k is the amount of secure storage. The protocol is based on a careful scrambling of a minimal set of server-hosted items. A partial reshuffle costing $O(n)$ is performed every time the secure storage fills up, which occurs once every k queries. While an improvement, this result is not always practical since the total database size n often remains much larger than the secure hardware size k. For $k = \sqrt{n}$ (as assumed in this paper), this mechanism yields an $O\left(\sqrt{n}\right)$ complexity (significantly greater than $O(\log\log n \log n)$ for practical values of n).

Fast ORAM

In [63], Williams et al. introduced a faster ORAM variant which also features correctness guarantees, with computational complexity costs and storage overheads of only $O(\log n \log\log n)$ (amortized per-query), under the assumption of $O\left(\sqrt{n}\right)$ temporary client storage. In their work, the assumed client storage is used to speed up the reshuffle process by taking advantage of the predictable nature of a merge sort on uniform random data.

Oblivious Transfer with Access Control

Camenisch et al. [24] study the problem of performing k sequential oblivious transfers (OT) between a client and a server storing N values. The work makes the case that previous solutions tolerate selective failures. A selective failure occurs when the server may force the following behavior in the ith round (for any i = 1...k): the round should fail if the client requests item j (of the N items) and succeed otherwise. The paper introduces security definitions to include the selective failure problem and then propose two protocols to solve the problem under the new definitions.

Coull et al. [29] propose an access control oblivious transfer problem. Specifically, the server wants to enforce access control policies on oblivious transfers performed on the data stored: The client should only access fields for which it has the credentials. However, the server should not learn which credentials the client has used and which items it accesses.

Recommended Reading

1. Activehost.com, Internet Services. Online at http://www.activehost.com
2. Adhost.com, MySQL Hosting. Online at http://www.adhost.com
3. Alentus.com, Database Hosting. Online at http://www.alentus.com
4. Datapipe.com, Managed Hosting Services. Online at http://www.datapipe.com
5. Discountasp.net Microsoft SQL Hosting. Online at http://www.discountasp.net
6. Gate.com, Database Hosting Services. Online at http://www.gate.com
7. Hostchart.com, Web Hosting Resource Center. Online at http://www.hostchart.com
8. Hostdepartment.com, MySQL Database Hosting. Online at http://www.hostdepartment.com/mysqlwebhosting/
9. IBM Data Center, Outsourcing Services. Online at http://www-1.ibm.com/services/
10. Inetu.net, Managed Database Hosting. Online at http://www.inetu.net
11. Mercurytechnology.com, Managed Services for Oracle Systems. Online at http://www.mercurytechnology.com
12. Neospire.net, Managed Hosting for Corporate E-business. Online at http://www.neospire.net
13. Netnation.com, Microsoft SQL Hosting. Online at http://www.netnation.com
14. Opendb.com, Web Database Hosting. Online at http://www.opendb.com
15. Amanatidis G, Boldyreva A, O'Neill A (2007) Provably-secure schemes for basic query support in outsourced databases. In: Barker S, Ahn G-J (eds) Data and applications security. Lecture notes in computer science, vol 4602. Springer, Berlin, pp 14–30
16. Asonov D (2004) Querying databases privately: a new approach to private information retrieval. Springer, Berlin
17. Bellare M, Boldyreva A, O'Neill A (2007) Deterministic and efficiently searchable encryption. In: Menezes A (ed) CRYPTO 2007. Lecture notes in computer science, vol 4622. Springer, Heidelberg, pp 535–552
18. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
19. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Cachin C, Camenisch JL (eds) Proceedings of EUROCRYPT 2004, Interlaken, 2–6 May 2004. Lecture notes in computer science, vol 3027. Springer, Heidelberg, pp 506–522
20. Boneh D, Gentry C, Lynn B, Shacham H (2003) Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham E

(ed) EUROCRYPT 2003. Lecture notes in computer science, vol 2656. Springer, Heidelberg, pp 416–432
21. Brinkman R, Doumen J, Jonker W (2004) Using secret sharing for searching in encrypted data. In: Proceedings of the secure data management workshop, Toronto, August 2004
22. Cachin C, Micali S, Stadler M (1999) Computationally private information retrieval with polylog communication. In: Stern J (ed) Proceedings of EUROCRYPT 1999. Lecture notes in computer science, IACR, vol 1592. Springer, Heidelberg. http://www.springerlink.com/content/0e5brpemrg9d5me7/
23. Cachin C, Micali S, Stadler M (1999) Computationally private information retrieval with polylogarithmic communication. In: Stern J (ed) Proceedings of EUROCRYPT 1999, Czech Republic, 2–6 May 1999. Lecture notes in computer science, vol 1592. Springer, Heidelberg, pp 402–414
24. Camenisch J, Neven G, Shelat A (2007) Simulatable adaptive oblivious transfer. In: EUROCRYPT '07: Proceedings of the 26th annual international conference on advances in cryptology, Barcelona, 20–24 May 2007. Lecture notes in computer science, vol 4515. Springer, Heidelberg, pp 573–590
25. Chang Y (2004) Single-database private information retrieval with logarithmic communication. In: Proceedings of the 9th Australasian conference on information security and privacy ACISP, Sydney, 13–15 July 2004. Lecture notes in computer science, vol 3108. Springer, Heidelberg, pp 50–61
26. Chang Y, Mitzenmacher M (2004) Privacy preserving keyword searches on remote encrypted data. Cryptology ePrint Archive, Report 2004/051. http://eprint.iacr.org/
27. Chor B, Goldreich O, Kushilevitz E, Sudan M (1995) Private information retrieval. In: IEEE symposium on foundations of computer science, Milwaukee, 23–25 Oct 1995. IEEE Computer Society, pp 41–50
28. Chor B, Kushilevitz E, Goldreich O, Sudan M (1998) Private information retrieval. J ACM 45(6):965–981
29. Coull S, Green M, Hohenberger S (2009) Controlling access to an oblivious database using stateful anonymous credentials. In: International conference on practice and theory in public key cryptography (PKC), Irvine, 18–20 Mar 2009. Lecture notes in computer science, vol 5443. Springer, Heidelberg, pp 501–520
30. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS '06: proceedings of the 13th ACM conference on computer and communications security, Alexandria, 30 October–3 November 2006. ACM Press, New York, pp 79–88
31. Devanbu PT, Gertz M, Kwong A, Martel C, Nuckolls G, Stubblebine SG (2001) Flexible authentication of XML documents. In: Proceedings of the ACM conference on computer and communications security, Philadelphia, 6–8 November 2001. pp 136–145
32. Devanbu PT, Gertz M, Martel C, Stubblebine SG (2000) Authentic third-party data publication. In: IFIP workshop on database security, Schoorl, The Netherlands, 21–23 Aug 2000. Kluwer, pp 101–112
33. Gartner, Inc. (1999) Server storage and RAID worldwide. Technical report, Gartner Group/Dataquest. www.gartner.com
34. Goh E (2003) Secure indexes. Cryptology ePrint Archive, Report 2003/216. http://eprint.iacr.org/2003/216/
35. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious ram. J ACM 45:431–473
36. Golle P, Staddon J, Waters B (2004) Secure conjunctive keyword search over encrypted data. In: Proceedings of ACNS, Yellow Mountain, 8–11 June 2004. Lecture notes in computer science, vol 3089. Springer, Heidelberg, pp 31–45
37. Hacigumus H, Iyer B, Li C, Mehrotra S (2002) Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM SIGMOD international conference on management of data, Madison, 3–6 June 2002. ACM Press, pp 216–227
38. Hacigumus H, Iyer BR, Mehrotra S (2002) Providing database as a service. In: IEEE international conference on data engineering (ICDE), San Jose, 26 Feb–01 Mar 2002, pp 29–38
39. Hore B, Mehrotra S, Tsudik G (2004) A privacy-preserving index for range queries. In: Proceedings of ACM SIGMOD. http://portal.acm.org/citation.cfm?id=1316752
40. Iliev A, Smith SW (2004) Private information storage with logarithmic-space secure hardware. In: Proceedings of i-NetSec 04: 3rd working conference on privacy and anonymity in networked and distributed systems, Toulouse, August 2004, pp 201–216
41. Kushilevitz E, Ostrovsky R (1997) Replication is not needed: single database, computationally-private information retrieval. In: Proceedings of FOCS, Miami Beach, 20–22 October 1997. IEEE Computer Society, pp 364–373
42. Kushilevitz E, Ostrovsky R (2000) One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In: Proceedings of EUROCRYPT, Bruges, 14–18 May 2000. Lecture notes in computer science, vol 1807. Springer, Heidelberg, pp 104–121
43. Lipmaa H (2004) An oblivious transfer protocol with log-squared communication. Cryptology ePrint Archive. http://www.springerlink.com/content/7n7219tcyxr81gxc/
44. Mann E (1998) Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology
45. Martel C, Nuckolls G, Devanbu P, Gertz M, Kwong A, Stubblebine S (2001) A general model for authenticated data structures. Technical report
46. Martel C, Nuckolls G, Devanbu P, Gertz M, Kwong A, Stubblebine SG (2004) A general model for authenticated data structures. Algorithmica 39(1):21–41
47. Merkle R (1980) Protocols for public key cryptosystems. In: IEEE symposium on research in security and privacy, Oakland, 14–16 April 1980
48. Motwani R, Raghavan P (1995) Randomized algorithms. Cambridge University Press, Cambridge, UK
49. Mykletun E, Narasimha M, Tsudik G (2004) Authentication and integrity in outsourced databases. In: ISOC symposium on network and distributed systems security NDSS. http://portal.acm.org/citation.cfm?id=1149977
50. Mykletun E, Narasimha M, Tsudik G (2004) Signature bouquets: immutability for aggregated/condensed signatures. In: Proceedings of the European symposium on research in computer security ESORICS, Sophia Antipolis, 13–15 September 2004. Lecture notes in computer science, vol 3193. Springer, Heidelberg, pp 160–176
51. Narasimha M, Tsudik G (2005) DSAC: integrity for outsourced databases with signature aggregation and chaining. Technical report
52. Narasimha M, Tsudik G (2006) Authentication of outsourced databases using signature aggregation and chaining.

In: Proceedings of DASFAA, Singapore, 12–15 April 2006. Lecture Notes in Computer Science, vol 3882. Springer, Heidelberg, pp 420–436
53. Paillier P (1999a) Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of EUROCRYPT, Prague, 2–6 May 1999. Lecture notes in computer science, vol 1592. Springer, Heidelberg, pp 223–238
54. Paillier P (1999b) A trapdoor permutation equivalent to factoring. In: PKC'99: proceedings of the second international workshop on practice and theory in public key cryptography, Kamakura, 1–3 March 1999. Springer, London, pp 219–222
55. Pang HH, Jain A, Ramamritham K, Tan KL (2005) Verifying completeness of relational query results in data publishing. In: Proceedings of ACM SIGMOD, Baltimore, June 2005, pp 407–418
56. Pang HH, Tan KL (2004) Authenticating query results in edge computing. In: ICDE '04: proceedings of the 20th international conference on data engineering, Boston, March/April 2004. IEEE Computer Society, Washington, DC, pp 560–571
57. Sion R (2005) Query execution assurance for outsourced databases. In: Proceedings of the very large databases conference VLDB, Trondheim, 30 August–2 September 2005, pp 601–612
58. Sion R, Carbunar B (2007) On the computational practicality of private information retrieval. In: Proceedings of the network and distributed systems security symposium, San Diego, 28 February–2 March 2007. Stony Brook Network Security and Applied Cryptography Lab Tech Report 2006-06
59. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: SP '00: proceedings of the 2000 IEEE symposium on security and privacy (S&P 2000), Berkeley, 14–17 May 2000. IEEE Computer Society, pp 44–55
60. Stern J (1998) A new and efficient all-or-nothing disclosure of secrets protocol. In: Ohta K, Pei D (eds) Proceedings of Asia crypt, Beijing, 18–22 October 1998. Lecture notes in computer science, vol 1514. Springer, Heidelberg, pp 357–371
61. Ge T, Zdonik S (2007) Answering aggregation queries in a secure system model. In: VLDB '07: proceedings of the 33rd international conference on Very large data bases, University of Vienna, Austria, 23–27 September 2007. VLDB Endowment, pp 519–530
62. Wang S, Ding X, Deng RH, Bao F (2006) Private information retrieval using trusted hardware. In: Proceedings of the European symposium on research in computer security ESORICS, Hamburg, 18–20 September 2006. Lecture notes in computer science, vol 4189. Springer, Heidelberg, pp 49–64
63. Williams P, Sion R, Carbunar B (2008) Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: ACM conference on computer and communication security CCS, Alexandria, 27–31 Oct 2008. ACM, New York, pp 139–148
64. Williams P, Sion R, Shasha D (2009) The blind stone tablet: outsourcing durability. In: Proceedings of the network and distributed systems security symposium NDSS, San Diego, 8–11 February 2009
65. Yang Z, Zhong S, Wright RN (2006) Privacy-preserving queries on encrypted data. In: Gollmann D, Meier J, Sabelfeld A (eds) ESORICS 2006, Hamburg, 18–20 September 2006. Lecture notes in computer science, vol 4189. Springer, Heidelberg, pp 479–495

Secure Device Pairing

Ming Li[1], Wenjing Lou[2], Kui Ren[3]
[1]Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
[2]Wireless Networking and Security, Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
[3]Ubiquitous Security & PrivaCy Research Laboratory (UbiSeC Lab), Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA

Synonyms

Secure first connect

Related Concepts

►Device Authentication; ►Diffie–Hellman Key Agreement; ►Hash Functions; ►Man-in-the-Middle Attack; ►Message Authentication Protocols

Definition

Secure device pairing is the process of bootstrapping a secure communication channel between two previously unassociated electronic devices communicating over some insecure channel.

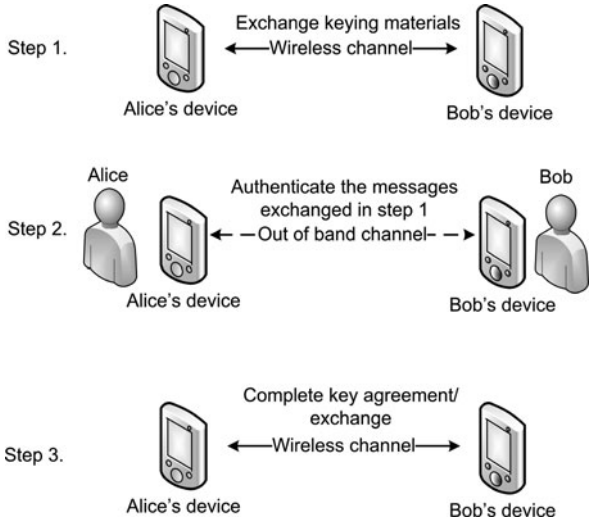
Background

The proliferation of portable electronic devices (e.g., personal digital assistants (PDAs), laptops, cell phones) has brought up numerous opportunities for communication from anywhere and at anytime. These devices are equipped with wireless communication capabilities and communicate with each other in an ad-hoc manner, i.e., communication can be setup without the help from any third party. Example applications include the use of Bluetooth to transfer data between two cell phones, associating a cell phone with the audio system in a car to make quick phone calls when driving, and using WiFi to associate a personal laptop to a 802.11 access point.

But opportunities always coexist with challenges. Since the personal devices are often low-cost and can be conveniently acquired by an individual, it is also easy for malicious people to gain access to them. To make things worse, the open wireless channel is easily subjected to various attacks such as eavesdropping, relaying, and forging, which poses serious threats for the security and privacy of ubiquitous communication.

Under such circumstances, one question arise naturally: how can two pervious unknown wireless devices establish initial trust in each other and set up a secure communication channel? Or, how can a user be assured that his/her device is communicating with exactly and uniquely the device held by the other person, but not some attacker's device nearby? This problem is particularly challenging. Building a public key infrastructure (PKI) is not a practical solution since it is basically infeasible (or of very high cost if possible) to centrally manage a huge amount of wireless devices. Involving an online trusted third party in the communication between any two ad hoc devices is again not viable. Therefore, there may exist no prior security context (including common secrets or common history) between two unfamiliar wireless devices.

Fortunately, secure device pairing is a new concept proposed to solve the problem of establishing a secure channel during the first connect between two unfamiliar wireless devices. The very core idea of secure device pairing is to make use of human interactions to help with the device authentication process. As a well-known example, in the Bluetooth protocol [5] for pairing two cell phones, a simple Personal Identification Number (PIN) is displayed on the screen of one phone and a user manually copies it to the other phone. Unlike wireless signals, the information transferred under the help of human interactions are perceivable by human. Thus, in device pairing, the human interaction is often employed as an auxiliary out-of-band (OOB) channel which is more secure than the wireless channel. In this way, the messages exchanged over wireless channels are authenticated by messages sent in OOB channels. Subsequently, a secret session key can be established between two unfamiliar devices. The idea of using OOB channel in device pairing is illustrated in Fig. 1.



Secure Device Pairing. Fig. 1 Graphical illustration of the general idea of secure device pairing (two user setting). Step 2 may or may not involve human interactions. Step 3 corresponds to the derivation of a common session key

Theory

A secure device pairing scheme usually consists of two parts: a *cryptographic protocol* and a *device pairing method*. The former denotes the entire information exchanged during device pairing, while the latter means the pairing process from the user's point of view [8].

Cryptographic Protocols

Related Cryptographic Primitives

Collision-resistant hash functions: cryptographic hash functions which are pre-image resistant, second pre-image resistant, and collision-resistant. Examples include SHA and MD5.

Digest functions: a short universal hash, or digest of a message keyed by a random number [13].

Commitment schemes. A commitment scheme consists of two algorithms: commit and open. It should have two properties: (1) *hiding*: the committed value is hidden from the receiver until the sender reveals it. (2) *binding*: it is infeasible to find another message that commits to the same commitment.

Pairwise Device Pairing

When two unfamiliar devices meet for the first time, secure device pairing is invoked to bootstrap a secure communication channel, which is usually achieved via establishing a symmetric session key shared only between the two devices. In the seminal paper published by Stajano and Anderson in 1999 [17], "resurrecting duckling" was proposed, in which a "master" device generates a random session key and transfers it to a "slave" device using cable connections.

A classical way for two communication parties to establish a symmetric key without physical device contact is the Diffie–Hellman key agreement (DHKA). Denote the two communicating devices as A and B. The process of DHKA between A and B is sketched as follows: A and B both choose a common large prime p and a generator g, and each of them selects a random number a and b, respectively. Then they both publish their public numbers $X_a = g^a \bmod p$ and $X_b = g^b \bmod p$. Finally, A computes $(X_b)^a = g^{ab} \bmod p$ and B computes $(X_b)^a = g^{ab} \bmod p$ as the session key.

However, assuming that the wireless channel can be modeled as a Dolev-Yao channel, DHKA over the wireless channel is subjected to the Man-in-the-Middle (MitM) attack. Defeating this attack requires an authenticated channel which can be realized by pre-shared common secrets or a PKI; but neither could exist for two ad hoc devices. Therefore, in device pairing, human-perceptible auxiliary OOB channels are employed as alternatives. The MitM attack is detectable by human under OOB channels, and messages exchanged over the wireless channel can be authenticated by messages exchanged over an OOB channel.

The ways that human interactions are exploited in pairwise device pairing are divided into two categories: direct transfer of information from one device to another through an OOB channel (**C1**), or human-based comparison of two pieces of short information (**C2**). Category **C1** usually corresponds to the unidirectional authentication, while **C2** corresponds to the bidirectional authentication.

Secure device pairing benefits from the above, which are exactly the underlying ideas of the *message authentication protocols* [13]. In device pairing, the keying materials (such as the public numbers in DHKA) are simply regarded as the messages to be authenticated in the message authentication protocols. The keying materials are first exchanged over the wireless channel, and then authenticated over the OOB channel. Finally, a common session key is derived using DHKA or negotiated through public key encryption. (See Fig. 1)

The following is an overview of cryptographic constructions for message authentication protocols used in secure device pairing.

(I) Unidirectional message authentication. To achieve unidirectional message authentication, non-interactive authentication protocols are usually employed [15]. If A wants to authenticate itself to B, information is only transferred from A to B, but not vice versa. The common idea of the unidirectional message authentication protocols can be described as: first send the message over the wireless channel, and then send some digest of the message over an OOB channel. Only when the former is verified to match with the latter, the message is accepted as authenticated.

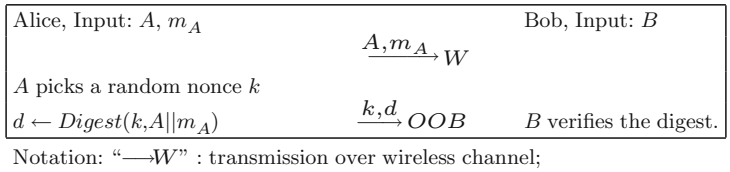
Examples. Early protocols use collision-resistant hash functions as the digest function [1, 15], which requires the digests to be at least 80 bits long and is not practical for human users. The MANA protocols [15, 19] use short authentication strings (SAS) as digests. The V-MANA I protocol is depicted in Fig. 2.

In the V-MANA I protocol, m_A is directly bound to a digest, which is sent along with a random number k in OOB channel. The OOB channel here is assumed to be strongly authenticated, i.e., it is stall-free, and messages cannot be replayed or deleted. Since k cannot be known in advance by an adversary, the adversary cannot perform any useful combinatorial search to find a collision. All that the adversary can do is to randomly guess a value of m'_A which digests to the same value as m_A, which succeeds with probability 2^{-b}, where b is the length of the digest. For practical security, b can be as small as 15 bits [19].

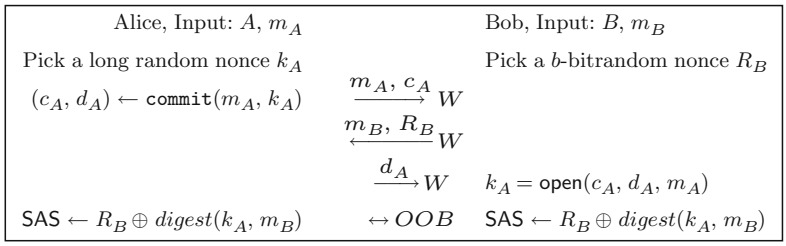
Other protocols adopt similar ideas to the above protocol, where an important problem is to make the protocol more efficient, i.e., transmit less bits in the OOB channel. For a comprehensive survey on message authentication protocols based on OOB channels, see [13].

(II) Bidirectional message authentication. To achieve bidirectional authentication, interactive protocols are often used, where both A and B exchange messages to be authenticated by each other. In general, since A and B has to receive the same set of messages, these protocols work by letting the users manually compare a short digest of the set of messages exchanged between A and B.

The underlying idea is "*commitment before knowledge*," i.e., to first commit the messages using some commitment scheme (via wireless channel), and reveal the messages after the commitments are received. Then, a digest that binds the protocol transcript is generated by each device, which is compared with each other through an



Secure Device Pairing. Fig. 2 The V-MANA I protocol [19]



Secure Device Pairing. Fig. 3 The Pasini–Vaudenay two-way authentication protocol [18]

OOB channel. Due to the hiding and binding properties of commitment schemes, upon reception of commitments, the committed messages are unknown to the adversary and cannot be changed. This precludes any offline attacks and limits the the adversary to randomly guessing which messages will yield a collision in the final digests.

Examples. The Pasini–Vaudenay two-way authentication protocol [18] is illustrated in Fig. 3. In this protocol, A commits its message m_A to B and binds it with a relatively long random number k_A. Thanks to the hiding and binding property of commitment scheme, k_A cannot be known in advance and cannot be manipulated. Since the digest contains k_A and m_B, and B's short random number R_B adds randomness to the digest, the success probabilities of random guessing k_A, m_A, and m_B are all limited to $\epsilon \leq 2^{-b}$. Note that, the OOB channel in this protocol is assumed to allow slight delay. On the definition and classification of the OOB channels, see [13].

Other examples include the Hoepman protocol [4], Cagalj–Capkun–Hubaux protocol [2], and the Laur–Nyberg pairwise protocol [10].

Extension to a Group of Devices

To bootstrap a secure channel among a group of N devices, simply doing pairwise paring between any two devices in the group takes $O(N^2)$ time, which is not efficient. In fact, using the same idea of "*commitment before knowledge*", pairwise device pairing can be extended to the group setting. Then, every device in the group needs to broadcast a commitment and compute a digest of all the messages generated by the group.

Detailed examples are not given here. Refer to the HCBK protocol [12] and the Laur–Pasini protocol [11].

Device Pairing Methods

For different applications, there are different device pairing methods, which are independent with and complement to the cryptographic protocols. Essentially, a pairing method is an implementation of the OOB channel, and its choice is closely related to the hardware capabilities of the devices. For example, visual channels can be adopted only if the devices have visual input and output interfaces.

Corresponding to protocols based on direct information transfer (**C1**), the "Seeing is Believing" [6] uses a visual channel to implement unidirectional authentication. With two high-end smart phones both equipped with camera and keyboards, one phone directly captures the screen output of the other's using a camera. For digest comparison-based protocols (**C2**), there are numerous pairing methods, such as "Blink-Blink" [16] which is based on synchronized audiovisual patterns, and "Loud-and-Clear" [3] which utilizes audio channel. For a survey on device pairing methods, refer to [7–9].

Applications

Secure device pairing has wide applications, especially in ubiquitous authentication in pervasive computing where no PKI or trusted party could exist. Examples include: exchanging data between two personal gadgets like PDA and cell phone, mobile payment from a credit card to a wireless card reader in a convenient store, setting up an ad hoc conference with each group member holding a PDA [14].

An important issue in applying device pairing schemes is the usability. The actual perceived security, pairing completion time, and error rates affect the wide acceptance of the pairing techniques and should be comprehensively evaluated. For a survey on usability studies of device pairing, refer to [7–9].

Recommended Reading

1. Balfanz D, Smetters DK, Stewart P, and Wong HC (2002) Talking to strangers: authentication in ad-hoc wireless networks. In: Proceedings of the 2002 Network and Distributed Systems Security Symposium (NDSS 2002), San Diego, 2002
2. Cagalj M, Capkun S, Hubaux JP (2006) Key agreement in peer-to-peer wireless networks. Proceedings of the IEEE (Special Issue on Cryptography and Security) vol 94(2), pp 467–478

3. Goodrich MT, Sirivianos M, Solis J, Tsudik G, Uzun E (2006) Loud and clear: human-verifiable authentication based on audio. International Conference on Distributed Computing Systems, Washington, 2006
4. Hoepman JH, Irda O (2004) The ephemeral pairing problem. In: Proceedings of the 8th International Financial Cryptography Conference, Springer, Berlin, pp 212–226
5. Jakobsson M, Wetzel S (2001) Security weaknesses in bluetooth. In: Proceedings of the 2001 Conference on Topics in Cryptology CT-RSA, London
6. McCune JM, Perrig A, Reiter MK (2005) Seeing-is-believing: Using camera phones for human-verifiable authentication. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, Oakland, pp 110–124
7. Kainda R, Flechais I, Roscoe AW (2009) Usability and security of out-of-band channels in secure device pairing protocols. In SOUPS '09: Proceedings of the 5th symposium on usable privacy and security, Mountain View, California, 2009
8. Kobsa A, Sonawalla R, Tsudik G, Uzun E, Wang Y (2009) Serial hook-ups: a comparative usability study of secure device pairing methods. In SOUPS '09: Proceedings of the 5th symposium on usable privacy and security, Mountain View, California, 2009
9. Kumar A, Saxena N, Tsudik G, Uzun E (2009) Caveat emptor: a comparative study of secure device pairing methods. In IEEE international conference on pervasive computing and communications (PerCom), Galveston, 2009
10. Laur S, Asokan N, Nyberg K (2005) Efficient mutual data authentication using manually authenticated strings. In: Cryptology and Network Security, Springer, Heidelberg, pp 90–107
11. Laur S, Pasini S (2008) SAS-Based Group Authentication and Key Agreement Protocols. In Public Key Cryptography – PKC 2008, LNCS, Barcelona, pp 197–213
12. Nguyen LH, Roscoe AW (2008) Authenticating ad hoc networks by comparison of short digests. Inf Comput 206(2–4): 250–271
13. Nguyen LH, Roscoe AW (2008) Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. J Comput Secur 195–210
14. Owen Chen CH, Chen CW, Kuo C, Lai YH, McCune JM, Studer A, Perrig A, Yang BY, Wu TC (2008) Gangs: gather, authenticate 'n group securely. In: Mobi- Com 2008, San Francisco, pp 92–103
15. Pasini S, Vaudenay S (2006) SAS-based authenticated key agreement. In: Public Key Cryptography – PKC '06, LNCS vol 3958, pp 395–409
16. Saxena N, Ekberg JE, Kostiainen K, Asokan N (2006) Secure device pairing based on a visual channel. In: IEEE Symposium on Security and Privacy, IEEE Computer Society, Oakland, May 2006
17. Stajano F, Anderson RJ (2000) The resurrecting duckling: security issues for ad-hoc wireless networks. In: Proceedings of the 7th international workshop on security protocols, Springer, pp 172–194
18. Sylvain Pasini and Serge Vaudenay. An optimal non-interactive message authentication protocol. Topics in Cryptology CT-RSA, 2006
19. Vaudenay S (2005) Secure communications over insecure channels based on short authenticated strings. In: Advances in cryptology – CRYPTO 2005. LNCS, vol 3621, Springer, Heidelberg, pp 309–326

Secure Element

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Related Concepts

►Common Criteria; ►HSM (= Hardware Security Module); ►Memory Card; ►NFC (= Near Field Communication); ►RFID; ►Root of Trust; ►SIM (= Subscriber Identity Module); ►Smart Card (= Chip Card); ►Token; ►TPM (= Trusted Platform Module)

Definition

A Secure Element is a hardware device component. It offers any number of the following non-exhaustive list of tamper resistant secure services to the rest of the device:

- Tamper detection
- Root of trust (e.g., start of an authentication chain)
- Secure memory (e.g., a Secure Element can store the private key of a public key pair)
- Cryptographically secure random number generation
- Cryptographic services (e.g., AES decrypting using a secret key stored in the Secure Element, signature of a message using a private key, verification of a signature)
- Secure generation of keys (e.g., generation of a public key pair or of a shared-secret key for an authentication)
- Secure monitoring of system resources (e.g., detection of hardware configuration changes)
- Secure execution of software modules (e.g., secure boot)
- Secure counting of events (e.g., usage counter for secret keys)
- Secure time measurements (e.g., for a time-bounded proximity detection protocol)
- Tamper resistant unique identifier (e.g., a unique serial number that cannot be forged)

"Secure Element" is a broad term; there exist many secure element embodiments: smart cards, SIM cards, memory cards, TPMs, etc. They also exist with various contact and contactless interfaces.

One standard possible way to rate the security level of a given secure element is to evaluate it according to Common Criteria. For example, modern smart cards are evaluated between EAL4 and EAL5+ following the Common Criteria terminology.

Background
Modern secure systems build trust relationships between the components. Such systems need a root of trust (i.e., a place where they can store the required secrets or where they can perform secure computations). It is known that intrinsically there is no way to build such a root of trust in software only; that is the reason why hardware secure elements have been introduced and are becoming ubiquitous nowadays.

Applications
Here are only a few examples: the SIM cards in mobile phones, the smart cards for payments, the TPMs in modern laptops, the smart cards for settop boxes, the HSMs used in banks to store the root keys, the smart cards for public transportation fare tickets, the RFIDs for the traceability of chirurgical instruments in hospitals, the secure microSD cards in mobile phones to use the mobile phone as a contactless payment card or access control token, etc.

Open Problems and Future Directions
The current trend is to see more and more standardization efforts toward secure elements interoperability: ISO standards, ECMA, ETSI, Global Platform standards, Common Criteria protection profiles, etc.

Recommended Reading
1. Common criteria portal, http://www.commoncriteriaportal.org/
2. Trusted computing group home page, http://www.trustedcomputinggroup.org/

Secure Email

►PEM, Privacy-Enhanced Mail

Secure Function Evaluation

►Multiparty Computation

Secure Hash Algorithm

►SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)

Secure Index

Gerardo Pelosi
Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano, Milano, Italy
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici (DIIMM), University of Bergamo, Dalmine (BG), Italy

Synonyms
Keyword-based retrieval over encrypted data

Related Concepts
►Asymmetric Encryption; ►Bloom Filters; ►Pseudo Random Function; ►Random Oracle Model Search on Encrypted Data; ►Searchable Symmetric Encryption; ►Secure Data Structures; ►Semantic Security

Definition
A secure index enables to check whether the content of semantically secure ciphertexts includes a specified keyword without applying decryption operations or revealing the keyword value. Secure indices are mainly motivated in an outsourcing scenario with multiuser settings, where encrypted documents and their indices are stored on a remote server and are frequently queried and updated. A secure index has to guarantee that the server learns nothing beyond the outcome of the sequence of search operations. Indeed, the honest-but-curious server may try to analyze the messages received during users' interactions in order to learn additional information on the content of the stored documents. When a collection of documents and its indices are accessed either by a single user or never updated, the most suitable solution is to maintain a local index with pointers to outsourced documents containing the searched keywords. In multiuser settings, the adoption of local indices shows clear difficulties in maintaining the consistency of local data structure across all users.

Background
The evolution of information and communication technologies (ICTs) allows users with either limited resources or limited expertise to rely on distributed services for storing large volumes of data reliably and for making it globally available at relatively low cost. In such a scenario, the service provider (SP) is relied upon only for the availability of data and the enforcement of basic access control mechanisms. The outsourced data is stored in encrypted form to secure its content from both malicious adversaries and the SP itself. Therefore, from the perspective of an authorized

user, the most useful and essential primitive to be executed on an outsourced data collection is a "searching" primitive. Ideally, the SP can execute the search of specific keywords on ciphertexts it stores without knowing the decryption key or the decrypted plaintext or the keyword value. Both the hosting server (i.e., the SP) and other malicious adversaries should learn no information beyond the outcome of the search operations. The applications that can benefit from the privacy guarantees provided by secure indices, spans from e-mail servers [1] to storage of sensitive data.

In general terms, the solutions for the problem of searching over encrypted data can be divided into two classes: symmetric and asymmetric encryption schemes.

In the setting of symmetrically-encrypted data, in a first stage, the user sends to the SP the collection of encrypted documents. In a second stage, the same user or a delegate (i.e., the one who possesses the encryption/decryption key) queries the SP in order to retrieve a subset of the outsourced documents. In [2] the keyword-based search over encrypted data was studied for the first time. In that paper, the authors proposed a scheme which encrypts each word of a document separately through a special two-layered encryption construction. The approach has a computational complexity linear in the size of the document collection and is vulnerable to statistical analyses of the distribution of the underlying plaintexts. In [3, 4], the authors associate an index to each document in the outsourced data collection and enable every user with the correct encryption key to compute a "trapdoor" corresponding to a keyword of her choice. The server is delegated to execute the searching operation on the secure indices using such a trapdoor. As a consequence, the computational complexity required for a retrieval request is linear in the number of documents included in the outsourced data collection. In [4], the author introduces a formal definition of a secure index, formulates a security model known as "semantic security against chosen-keyword attack" (IND-CKA), and puts forth a secure index construction based on Bloom filters and pseudo-random functions. In [3] the authors propose a simulation-based security definition (IND2-CKA), so that indices with different number of keywords cannot be distinguished, and put forth a modified index construction for a whole document collection in order to guarantee that the trapdoors do not leak any information about the keywords being queried.

Improved indistinguishability and simulation-based definitions for searching over encrypted data using secure indices are given in [5, 6]. In those papers, the authors define a better adversarial model where the attacker can choose her own queries to the index as a function of previously obtained trapdoors and search outcomes. The proposed secure index construction has a constant computational complexity and is secure under the new definitions.

A detailed description of this construction is reported in the next section.

All the previous constructions give up the privacy of the access pattern (i.e., the documents matching the query/trapdoors) for the sake of efficiency. Indeed, more general security guarantees imply less efficient solutions such as oblivious RAMs [7]. Oblivious RAMs hide all information about the RAM use from the hosting server. Unfortunately, they involve a computational and communicational poly-logarithmic overhead that increases over time, and a logarithmic number of client-server interactions for each memory operation.

In the public-key setting, the outsourced data is encrypted using the public-key of a user U. This means that anyone may encrypt some information for U (storing it on the SP), thus adding potentially new words to be indexed. In order to perform a keyword search, U generates a trapdoor information and sends this information to the SP. The knowledge of the trapdoor allows the SP to perform the desired operation while preventing him to gain any information on the keyword or on the information that is associated to it. The first scheme in this class has been presented in [1]. In this paper, the authors present a scheme that is secure in the random oracle model. Starting from [1], the authors of [8] prove that the existence of an anonymous identity-base encryption (IBE) scheme implies the existence of public-key searchable encryption schemes. Furthermore, the authors present a construction for an identity-based encryption scheme with keyword search based on the existence of anonymous hierarchical IBE schemes.

The security guarantees of public-key schemes are weaker than the ones of the best symmetric-key scheme, and also the efficiency of the available solutions is at most comparable with their symmetric-encryption counterparts.

Theory

In this section, we describe the searchable symmetric encryption (SSE) scheme defined in [5, 6], using the same notation and terminology introduced in [2–4].

Among the users who take part in an SSE scheme, there is a client who wishes to encrypt a document collection $D = (D_1, \ldots, D_n)$ and store it on a remote server, while maintaining the functionality of finding all documents containing a particular keyword. A secure index I is

associated to D, and every user with the correct encryption key is able to compute a "trapdoor" for a specified keyword. The trapdoor allows the server to help in the retrieval of documents, but prevents him to know every additional information on the accessed data beyond the outcome of the search operation and any information inferred from the a-priori knowledge that two or more searches were performed for the same keyword or not. The formal transposition of such a requirement is defined as "Adaptive Indistinguishability Security" for SSE schemes [6] and is employed to prove that the encrypted documents, the index, and the trapdoors involved in any two adaptively-constructed searches cannot be distinguished (i.e., correlated or predicted) with probability non-negligibly better than 0.5 by any polynomially bounded adversary who makes search queries as a function of the outcome of previous searches. Let *SSE*=(*Keygen, BuildIndex, Trapdoor, Search*) be a collection of four polynomial-time algorithms. *Keygen*(k) is a key generation algorithm run in order to set up the scheme. It takes a security parameter k and returns a secret key K with length polynomially bounded in k. *BuildIndex*(K, D) is the index generation algorithm. It takes a secret key K and a set of documents D in order to return a secure index I with size polynomially bounded in k. *Trapdoor*(K, w) is the algorithm run to generate a trapdoor T_w for a given secret key K and a given word w. *Search*(I, T_w) is run by the hosting server in order to search for the documents in D that contain word w. It takes the index I for the collection D and a trapdoor T_w for word w as inputs, and returns a lexicographically sorted list of document identifiers containing w, denoted as:

$$D(w) = \{id(w,1), \ldots, id(w,j), \ldots\}, \text{ with } 1 \le j \le |D(w)|$$

Defined as "unit" the smallest possible size for a word (e.g.: one byte), in the following, we assume l be the size of the largest plaintext document in D (expressed in units), n the number of documents in D, and $m = l \cdot n$ an upper bound to the total size of the plaintext document collection. We denote the set of plaintext words in D as Δ, and the corresponding subset of distinct plaintext words as $\Delta' \subseteq \Delta$.

We also let $\texttt{label}_{w,j} \in \{0,1\}^p$ be the binary string resulting from the concatenation of a word $w \in \Delta'$ with a string representing the ordinal number of an identifier in $D(w)$, $1 \le j \le |D(w)|$, and let π: $\{0,1\}^k \times \{0,1\}^p \to \{0,1\}^p$ be a pseudo-random permutation function.

The enforcement of the *BuildIndex*(K, D) primitive [5, 6] computes the list of document identifiers $D(w)$ for each $w \in \Delta'$ and builds the secure index I as a look-up table with m ($\le 2^p$) entries. Each entry is described by a pair (address, value), referred as I[address] ← value, and is generated as follows:

$$I[\pi(K, \texttt{label}_{w,j})] \leftarrow id(w,j) \text{ for each } w \in \Delta' \text{ and } 1 \le j \le |D(w)|$$

To hide the number of distinct words in each document, given $m' = \Sigma_{w \in \Delta'} |D(w)|$, if $m' < m$, the $m - m'$ entries of I must be filled so that the identifier of each document appears in the same number of entries.

The *Trapdoor*(K, w) primitive is run by the user to generate a trapdoor for a given word w as following:

$$T_w \leftarrow (t_1, \ldots, t_l) = (\pi(K, \texttt{label}_{w,1}), \ldots, \pi(K, \texttt{label}_{w,l}))$$

Finally, the *Search*(I, T_w) primitive consists of l lookups since each of the trapdoor components "reveals" only one entry in the index.

Despite being remarkably simple, the described SSE scheme is proven to be secure, and requires for each query: one round of communication, constant storage and computational costs on the user side, and a computational complexity proportional with the number of documents that match the query, on the server side. Finally, the improved primitives to deal with the multiuser setting [6], where an arbitrary group of parties other than the content owner can submit search queries, employ well-known techniques for "broadcast encryption" without modifying the basic scheme previously described.

Also, the optimizations for the secure index updates [6] follow the approach in [3] and do not modify the described mechanisms.

Recommended Reading

1. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Cachin C, Camenisch J (eds) Proceedings of the international conference on the theory and applications of cryptographic techniques. Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer, Berlin, pp 506–522
2. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE symposium on security and privacy (SP '00). IEEE Computer Society, Washington, DC, USA, pp 44–55
3. Chang Y-C, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. In: Ioannidis J, Keromytis, Moti Yung (Eds) Proceedings of the third applied cryptography and network security conference (ACNS 2005). Lecture Notes in Computer Science, vol 3531. Springer, pp 442–455
4. Goh E-J (2003) Secure indexes. Cryptology ePrint archive, report 2003/216. http://eprint.iacr.org/2003/216/
5. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on computer and communications security (CCS '06). ACM, New York, pp 79–88

6. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. Cryptology ePrint archive, report 2006/210. http://eprint.iacr.org/2006/210
7. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious RAMs. J ACM 43(3):431–473
8. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2005) Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup V (ed) Proceedings of the 25th annual international cryptology conference. Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol 3621. Springer, Berlin, pp 205–222

Secure Localization

Donggang Liu
Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas, USA

Synonyms

Secure location discovery

Related Concepts

►Localization; ►Security Protocol

Definition

A secure localization protocol is one that correctly estimates the locations of sensor nodes in the presence of malicious attacks.

Background

Sensors' locations play a critical role in sensor network applications. Not only do applications such as environment monitoring and target tracking require sensors' location information to fulfill their tasks, but several fundamental techniques developed for wireless sensor networks also require sensors' locations. For example, in many geographical routing protocols, sensor nodes make routing decisions at least partially based on their own and their neighbors' locations. The correctness of location information is critical for these applications and techniques to function correctly.

Sensors' locations are often estimated based on certain measurements (e.g., distance and angle) with respect to multiple known positions in the network by observing features like received signal strength (RSS) or time difference of arrival (TDoA). For example, some *beacon nodes* who knew their locations may periodically broadcast beacon packets for other sensors to estimate the distances to them based on the received signal strength. As long as the distances to three known positions are measured, triangulation can be used to estimate the sensor location. However, in the presence of malicious attacks, a sensor node may get incorrect information, leading to an incorrectly estimated location. Examples of malicious attacks against localization are given in Fig. 1.

Theory

Many techniques have been developed recently to ensure the security of localization in sensor networks. These techniques provide secure localization by (i) protecting the measuring of physical signal features, (ii) tolerating malicious measurements, or (iii) detecting malicious nodes that supply misleading beacon signals.

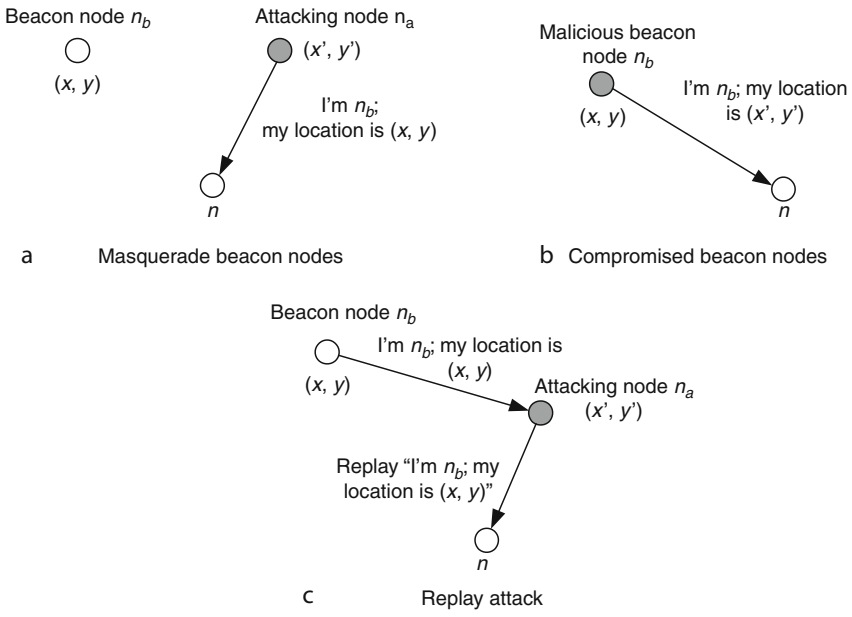
Distance Bounding-Based Schemes

A distance bounding protocol is a security protocol that establishes an upper bound for the distance between two communicating parties; it was first introduced in [1]. The main idea is based on the fact that light travels in a finite speed. Once the time needed to propagate packets over the air is measured, it is easy to estimate the upper bound on the distance between the sender and the receiver. Such upper bound can then be used to improve the accuracy of localization. The distance bounding has been suggested for protecting localization in many studies [2–4]. However, it requires nanosecond processing and time measurements, which are often not available in the sensor network.

Attack-Resistant MMSE

The attack-resistant minimum mean square estimation (MMSE) technique is a secure localization protocol that can correctly estimate sensors' locations even if some of the *location references* have been manipulated by adversaries [5, 7]. A location reference is a three tuple $\langle x, y, \delta \rangle$, where (x, y) represents the location of the beacon node and δ is the measurement (e.g., distance) with respect to this beacon node. This technique is based on the fact that malicious location references are introduced to mislead a sensor node about its location, and thus are usually inconsistent with benign location references. Therefore, the main task is to check the inconsistency among location references (indicated by the mean square error of estimation) and defeat attacks by removing such malicious data. One example of such consistency can be defined as follows: a set of location references $\mathcal{L} = \{ \langle x_1, y_x, \delta_1 \rangle, \langle x_2, y_2, \delta_2 \rangle, \cdots, \langle x_m, y_m, \delta_m \rangle \}$ obtained at a sensor node is *τ-consistent w.r.t an MMSE-based method* if the method gives an estimated location $(\cap x, \cap y)$ such that the mean square error of this estimation

$$\varsigma = \sum_{i=1}^{m} \frac{(\delta_i - \sqrt{(\cap x - x_i)^2 + (\cap y - y_i)^2}}{m} \leq \tau^2.$$



Secure Localization. Fig. 1 Attacks against location discovery schemes

Given the above definition about the consistency, it is natural to identify the largest consistent set from a set of location references collected at a sensor node. In [7], three methods are proposed to identify the largest consistent set, the brute-force algorithm (BARMMSE), the greedy algorithm (GARMMSE), and the enhanced greedy algorithm (EARMMSE).

Detection of Malicious Beacons

In hostile environments, a compromised beacon node may give malicious beacon signals that include incorrect locations, or manipulate beacon signals so that a receiving node obtains, for example, an incorrect distance measurement. Sensor nodes that use such beacon signals for localization will estimate incorrect locations. In [6], an detector was proposed to catch beacon nodes supplying malicious beacon signals. The basic idea is to deploy additional *witness* nodes to monitor the beacon signals. The witness nodes know their own locations in the field. Once a witness node receives a beacon signal, it obtains the location reference $\langle x, y, \delta \rangle$. Suppose that the distance measurements are used. The witness node then also compute the distance from (x, y) and its own location, and compare such computed distance with δ. If two distances are significantly different from each other, the beacon node is very likely a compromised node.

Applications

Secure localization has many applications in sensor network security protocols. For example, in some routing protocols, sensors' location information is directly used to make routing decisions. As another example, some clustering algorithms uses the sensors' location to form clusters. As a result, the correctness of location information directly impacts the correctness of these protocols.

Open Problems

One fundamental problem for secure localization is how to correctly estimate certain measurement (e.g., distance) to a reference point. Such estimation is usually based on certain physical features (e.g., received signal strength) observed by a given node. Existing solutions assume that the majority of the measurements obtained by a given node are correct. When this assumption does not hold, they will not be able to give correct location estimation. An open problem will be how to check whether a given measurement, estimated from certain physical features, can be trusted even if the majority of measurements are incorrect.

Recommended Reading

1. Brands S, Chaum D (1994) Distance-bounding protocols. In: Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, pp 344–359. Springer, New York
2. Capkun S, Hubaux JP (2005) Secure positioning of wireless devices with application to sensor networks. In: Proceedings of IEEE InfoCom, Miami, Florida

3. Lazos L, Capkun S, Poovendran R (2005) Rope: Robust position estimation in wireless sensor networks. In: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN), Nashville, Tennessee
4. Lazos L, Poovendran R (2004) Serloc: secure range-independent localization for wireless sensor networks. In: ACM workshop on Wireless security (ACM WiSe 2004), Philadelphia, Pennsylvania October 1 2004
5. Liu D, Ning P, Du WK (2005) Attack-resistant location estimation in wireless sensor networks. In: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN), Nashville, Tennessee
6. Liu D, Ning P, Du WK (2005) Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In: Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS), Colombus, Ohio
7. Liu D, Ning P, Liu A, Wang C, Du WK (2008) Attack-resistant location estimation in wireless sensor networks. ACM Transactions in Information and Systems Security (TISSEC)

Secure Location Discovery

►Secure Localization

Secure Logging

►Secure Audit Logs

Secure Multiparty Computation

►Multiparty Computation

Secure Multiparty Computation (SMC)

Keith B. Frikken
Department of Computer Science and Software Engineering, Miami University, Oxford, OH, USA

Synonyms

Multiparty computation (MPC)

Related Concepts

►Privacy-Preserving Protocols

Definition

Secure Multiparty Computation refers to cryptographic protocols that allow for the distributed computation of a function over distributed inputs without revealing additional information about the inputs.

Background

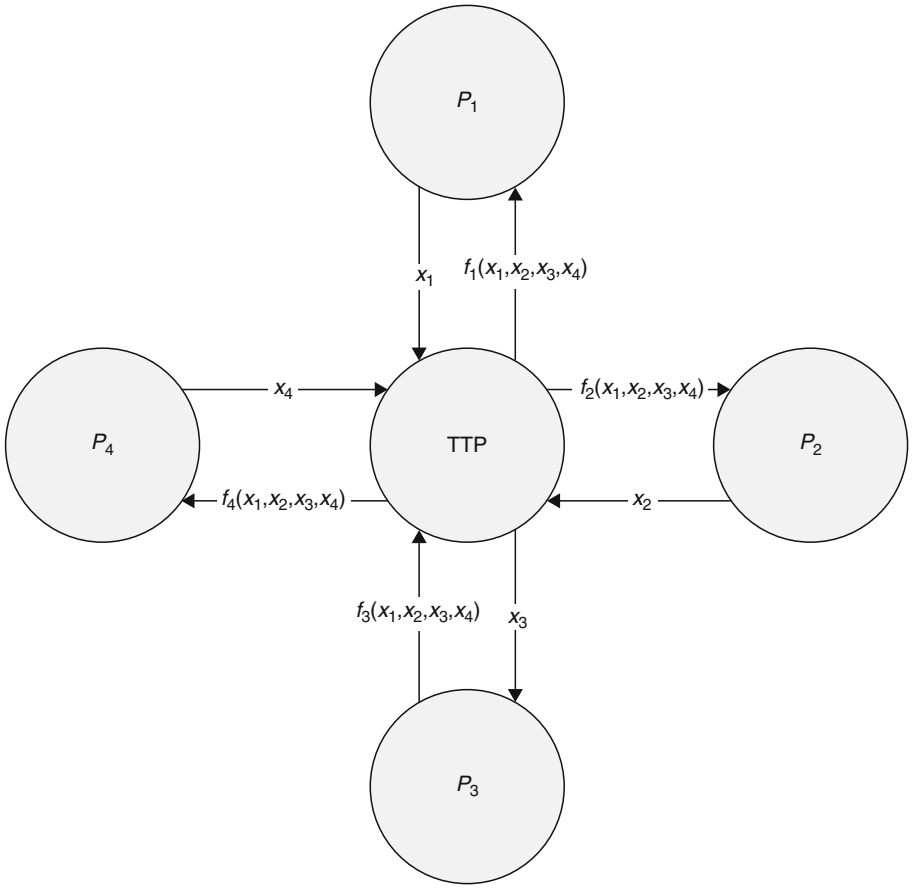
Yao [1] introduced the idea of secure multiparty computation by introducing the Millionaire's problem, which allowed two parties to determine whose value was larger without revealing anything else about the individual values.

Theory

In this setting, there are n parties, denoted by $P_1, \ldots, P_n$ with respective inputs $x_1, \ldots, x_n$. There is also a set of publicly known functions $f_1, \ldots, f_n$ each over the n inputs. At the end of the protocol, each participant obtains the value of its corresponding function, that is participant P_i obtains the value $f_i(x_1, \ldots, x_n)$. The security goal is that each participant should not learn anything else other than what can be inferred from x_i and $f_i(x_1, \ldots, x_n)$. Furthermore, if a set of parties, $C \subseteq \{P_1, \ldots, P_n\}$ collude with each other that nothing should be revealed to this group other than what can be inferred from $\{(x_i, f_i(x_1, \ldots, x_n)) : P_i \in C\}$. Secure Multiparty Computation (SMC) refers to a protocol for computing these functions while achieving this security goal.

Protocols for SMC are often compared to an ideal protocol that utilizes a trusted third party. A trusted third party (TTP) is an additional party that all participants fully trust. If such a party exists, then it is trivial to solve all SMC problems, because each party can send their input to the TTP who then sends the results to each party (see Fig. 1). Clearly, it is unrealistic to assume that such a party exists; however, the security goal of SMC protocols is to achieve the security of this TTP protocol without using the TTP.

There are two classic adversary models for SMC: honest-but-curious and malicious. In the honest-but-curious adversary model, the adversary controls a subset of the parties, but will follow the protocol faithfully. In this model, the adversary's goal is to learn additional information about the inputs of the honest parties. In the stronger malicious adversary model, the adversary will deviate from the protocol arbitrarily. The goal of this adversary is to either learn information about the inputs of the honest parties or to modify the output results of the honest parties. Furthermore, many SMC results assume that the adversary is computationally bounded (i.e., that the adversary is a probabilistic polynomial time algorithm). Putting these



Secure Multiparty Computation (SMC). Fig. 1 Ideal protocol with four parties

pieces together an SMC protocol is secure, against a specific adversary model, if there is another adversary that can achieve a similar effect in the protocol that utilizes a TTP.

While the notion of security achieved by SMC protocols is strong, there are several general results that state that this type of security can be achieved for any function. More specifically, general SMC protocols exist for computationally unbounded adversaries in the following cases [2–4]:

1. Honest-but-curious adversaries that corrupt any number of parties
2. Assuming a broadcast channel, malicious adversaries that corrupt at most a strict minority of the parties (i.e., more than half of the parties must be honest)
3. Assuming a broadcast channel, malicious adversaries that corrupt any number of parties, as long as it is not considered a security failure if the protocol terminates early

Applications

The general results of SMC were once considered theoretical and impractical. However, recent implementations of these results, such as the Fairplay system [5], have demonstrated that the results were practical for some problems. Furthermore, SMC has been used in a large-scale deployment of SMC that was used in a Sugar Beet Contract Auction [6].

Recommended Reading

1. Yao AC (1982) Protocols for secure computations (extended abstract). In: Proceedings of the 23rd annual IEEE symposium on foundations of computer science, Chicago, pp 160–164
2. Yao AC (1986) How to generate and exchange secrets. In: Proceedings of the 27th annual IEEE symposium on foundations of computer science, IEEE Computer Society, Washington, DC, pp 162–167
3. Goldreich O, Micali S, Wigderson A (1987) How to play ANY mental game. In: Proceedings of the 19th annual ACM conference on theory of computing, New York, pp 218–229
4. Goldreich O (2004) Foundations of crytpography: volume II basic applications, Cambridge University Press, Cambridge

5. Malkhi D, Nisan N, Pinkas B, Sella Y (2004) Fairplay – a secure two-party computation system. In: Proceedings of the 13th conference on USENIX security symposium, San Diego, pp 287–302
6. Bogetoft P, Christensen D, Damgrd I, Geisler M, Jakobsen T, Krigaard M, Nielsen J, Nielsen J, Nielsen K, Pagter J, Schwartzbach M, Toft T (2009) Secure multiparty computation goes live. In: Financial cryptography and data security, Springer, Berlin, pp 325–343

Secure Network Coding for Wireless Mesh Networks

Cristina Nita-Rotaru[1], Reza Curtmola[2]
[1]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA

Synonyms

Secure wireless mesh networks

Definition

Secure network coding for wireless mesh networks refers to protocols and mechanisms design to achieve reliable and secure communication in wireless mesh networks via network coding, resilient to inside and outside attacks. In a more limited sense, it refers to network coding protocols that can tolerate pollution attacks.

Background

Network coding is a promising paradigm that has been shown to improve throughput and provide elegant solutions to problems that were traditionally considered difficult, such as congestion control and reliability. The core principle of network coding is that intermediate nodes actively mix (or *code*) input packets and forward the resulting coded packets.

Several practical systems have been proposed to bridge theory with practice in the context of wireless mesh networks [3, 5, 13, 16, 20, 21]. Such systems need to solve a plethora of practical aspects before network coding meets its promised potential. To cope with the unpredictable and challenging wireless environment, network coding systems make practical design choices and optimizations that are essential to leverage network coding and achieve good performance. However, many of these design choices result in protocols that have numerous security vulnerabilities.

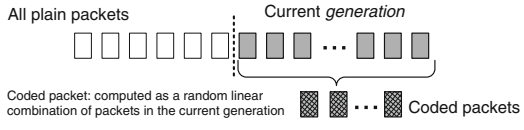
Network coding systems proposed for wireless networks leverage the benefits of network coding in two different ways: They mix packets within the same individual flow (referred to as *intra-flow* network coding systems), or they mix packets across multiple different flows (referred to as *inter-flow* network coding systems). Examples of intra-flow network coding systems include [3, 20, 21] and examples of inter-flow network coding systems include [5, 13, 16].

Intra-flow Network Coding

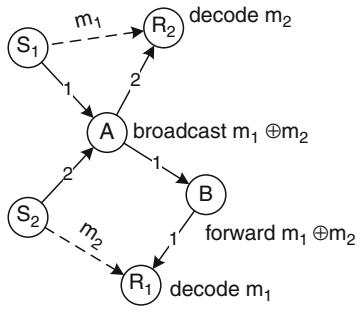
In intra-flow network coding systems, packets are delivered in batches, referred to as *generations*. Each node forwards *coded packets* which are computed as a random linear combination of the plain packets in a generation (see Fig. 1). To send a generation of packets, the source node continuously broadcasts coded packets for the current generation until an acknowledgment (ACK) is received from the destination. The coded packets are relayed to the destination via a set of intermediate nodes, referred to as *forwarder nodes*. Each forwarder node stores linearly independent coded packets it overhears and forwards new coded packets by combining the coded packets stored in its buffer. When the destination node receives enough linearly independent coded packets, it decodes the packets by solving a system of linear equations and unicasts an ACK packet to the source node, allowing the source to start sending the next generation of packets.

Inter-flow Network Coding

Inter-flow network coding exploits opportunistic listening and wireless broadcast with *opportunistic coding* at intermediate nodes. When a node has a set of packets for different flows to be delivered to different next hop nodes, instead of unicasting each packet individually to its corresponding next hop node, the node combines the packets together and broadcasts the combined packet once for all the next hop nodes. Therefore, inter-flow coding reduces multiple individual unicast transmissions to only one broadcast transmission. Inter-flow network coding systems are generally designed on top of traditional routing protocols. Given a set of paths for different flows, inter-flow network coding identifies a set of nodes at the intersections



Secure Network Coding for Wireless Mesh Networks. Fig. 1 An illustration of plain packets, generation, and coded packets in intra-flow network coding systems



Secure Network Coding for Wireless Mesh Networks. Fig. 2 An illustration of coding and decoding in an inter-flow network coding system. The number on edges represents the flow ID that the edge is on. Dashed lines represent packet overhearing

of paths to combine unicast transmissions of plain packets for different flows into broadcast transmissions of coded packets. The downstream nodes decode the coded packets using their overheard packets (see Fig. 2). Thus, unlike intra-flow network coding where all forwarder nodes perform coding and only receiver nodes perform decoding, in inter-flow network coding only nodes at the intersections of flows perform coding operations, and any downstream nodes that have overheard necessary packets can perform the decoding operation.

Theory

Attacks against Network Coding

Adversarial Model

A network coding system may come under attack from both *outside* and *inside* attackers. Outside attackers are limited to attacking the network without having access to network resources; this includes eavesdropping, injection, modification, and replay of packets. Inside attackers, having obtained access to authentication material (e.g., cryptographic keys), can pose as authorized participants and attack the network from the inside. Inside attackers are either nodes that have been compromised or honest nodes that have turned malicious.

Intra-flow Network Coding

An intra-flow coding system consists of the following components: Forwarding node selection and rate assignment, data forwarding, acknowledgment delivery, and packet coding/decoding. Each of these components can be subjected to attacks.

The forwarding node selection and rate assignment process relies on a link state graph, which is obtained by having each node monitoring its local link qualities, and periodically flooding the information in the entire network. An attacker can influence the node selection by claiming false metrics for its own adjacent links or by modifying link qualities reported by other nodes as they are flooded in the network. Link falsification attacks are difficult to prevent as this information is local to the attacker node. Link modification attacks can be prevented by using message authentication.

Wormhole attacks introduce fictitious links between honest nodes and distort their perception of network topology. Although existing wormhole solutions, such as packet leashes [11] and TrueLink [8] can be applied, they typically incur substantial overhead, which can potentially nullify the performance gain of network coding.

The data forwarding component is vulnerable to packet pollution and packet dropping attacks. In packet pollution attacks, the attacker node injects corrupted packets into the network (i.e., packets that are not linear combinations of the plain packets at the source). As each forwarder node combines received packets to form new coded packets, pollution attacks can cause an epidemic effect, where the corrupted packets from one affected honest node further affect other honest nodes. By injecting even a few corrupted packets, the attacker can degrade the performance significantly. In packet dropping attacks, an attacker arbitrarily drops data packets. Addressing packet dropping attacks in network coding systems is more challenging than in traditional routing protocols, as the number of packets a node transmits and the time of transmissions are contingent on the opportunistic receptions at the node.

The acknowledgment delivery component is vulnerable to attacks where an attacker injects, modifies, drops, or delays ACKs with severe consequences for the system. Some of these attacks can be prevented by using authentication mechanisms, while others, such as the ACK delaying attack, are stealthy and are more difficult to defend against.

Inter-flow Network Coding

An inter-flow coding system consists of the following components: Discovery of coding opportunities, packet coding/decoding, packet forwarding, and routing integration for increased coding opportunities.

The correct discovery of coding opportunities at a node relies on the correct packet reception information that the node collects from other nodes. An attacker can impersonate honest nodes and report incorrect packet reception information to their neighbors. Such an attack can cause a node to send coded packets that cannot be decoded by the intended next hop nodes. Since such non-decodable packets cannot be acknowledged, it will cause the sender

node to continuously transmit useless packets. This attack can be addressed with message authentication schemes. Attacks on link state routing protocols which cause a node to derive an incorrect link state graph can also cause a node to infer incorrect packet reception information and consequently send non-decodable packets. In systems that determine coding opportunities based on the neighboring node set information collected during the route discovery process, an attacker can cause neighbor set pollution either by direct modifications of route request packets or by using wormholes to introduce fictitious links. Existing approaches for secure source routing protocols and for defending against wormhole attacks can be used to secure the neighbor set information.

The packet transmission process in inter-flow coding systems is also subject to attacks such as ACK injection or modification, packet pollution, packet under- or over-coding, and packet dropping. Out of these, packet over- or under-coding are specific to inter-flow coding systems, while the others are similar to the ones in intra-flow coding systems with the caveat that they are more difficult to defend against. For example, existing pollution defense schemes proposed for intra-flow coding, such as homomorphic signatures, cannot be applied in inter-flow coding systems, as coded packets are formed from packets generated by different sources. Traditional monitoring mechanisms where an upstream node, say A, monitors the correct behavior of a downstream node, say B, by comparing the packets sent by A and those forwarded by B are also not effective under inter-flow network coding since an upstream node usually cannot decode the packets sent by a downstream node.

During the coding process, a malicious node can also conduct under- and over-coding attacks by coding less or more packets than it is supposed to. For example, consider a malicious coding node that is at the intersection of three flows and is supposed to code packets from two of the three flows; however, the node codes together packets from all the three flows, causing the downstream nodes unable to perform the decoding. The difficulty of defending against such attacks comes from the fact that the packets forwarded by the attacker do not match the definition of a corrupted packet. Thus, even if a defense scheme was able to detect polluted packets, it would still not be effective against packet under- or over-encoding attacks.

In order to select an optimal route that considers coding opportunities, a coding-aware routing protocol not only requires the correctness of link and path metrics as in traditional routing protocols, but it also requires the correctness of coding benefits reported by each node. Thus, in addition to manipulating link and path metrics, an attacker node can disrupt the protocol by manipulating coding opportunities. For example, by reporting very high coding opportunities, the attacker can improve its chances to be selected on the path and gain the control over the flow. Since coding opportunities not only depend on the network topology, but also depend on the current flow structure, it is more challenging to ensure the correctness of coding opportunities reported by a node than ensuring the correctness of pure topological metrics (e.g., link or path qualities).

Defenses

The only attacks studied are pollution attacks in intra-flow network coding systems. Current solutions to packet pollution attacks in intra-flow coding systems can be categorized into cryptographic approaches and information theoretic approaches.

Cryptographic approaches rely on augmenting the network coded packets with additional verification information; this allows intermediate nodes to verify the validity of coded packets and filter out polluted packets. Existing schemes use specialized homomorphic hash functions or homomorphic digital signatures. In hash-based schemes [9, 14, 15], the source uses a homomorphic hash function to compute a hash of each native data packet and sends these hashes to intermediate nodes via an authenticated channel. Most of the schemes based on digital signatures [4, 17, 18, 22] require reliable distribution of a new public key for every new file that is sent, and the size of the public key is linear in the file size. The only exception is a recent scheme [2], which achieves constant-size public key at the cost of using bilinear maps. Several schemes avoid the cost of using heavy-weight cryptographic primitives. For example, by assuming the existence of loose time synchronization in the network, [6] proposes a lightweight solution based on efficient linear checksums and time asymmetry. The scheme presented in [19] avoids using homomorphic signatures or hashes all together by relying solely on much more efficient symmetric key encryptions; however, the drawback is the significantly larger bandwidth overhead. Finally, the scheme in [1] uses efficient homomorphic MACs, but requires key pre-distribution from a centralized trusted entity, and its security decreases as the number of compromised nodes increases.

Information theoretic approaches do not filter out polluted packets at intermediate nodes; instead, they either encode enough redundant information into packets which allows receivers to detect the presence of polluted packets [10] or use a distributed protocol which allows receivers to tolerate pollution and recover native packets [12]. However, given that polluted packets are not filtered

out, the throughput that can be achieved by the protocol is upper-bounded by the information-theoretic optimal rate of $C - z_O$, where C is the network capacity from the source to the receiver and z_O is the network capacity from the adversary to the receiver.

Applications

It was shown that network coding provides benefits such as increased throughput, reduced network congestion, increased reliability, and reduced power consumption. Examples of wireless applications using secure network coding include: content distribution, storage, multicast, and network monitoring.

Recommended Reading

1. Agrawal S, Boneh D (2009) Homomorphic macs: mac-based integrity for network coding. In: Proceedings of ACNS '09, Paris-Rocquencourt
2. Boneh D, Freeman D, Katz J, Waters B (2009) Signing a linear subspace: signature schemes for network coding. In: Proceedings of PKC '09, LNCS vol 5443, Springer, Heidelberg, pp 68–87
3. Chachulski S, Jennings M, Katti S, Katabi D (2007) Trading structure for randomness in wireless opportunistic routing. In: Proceedings of ACM SIGCOMM '07, San Diego
4. Charles D, Jain K, Lauter K (2006) Signatures for network coding. In: Proceedings of CISS '06, Princeton, pp 857–863
5. Das S, Wu Y, Chandra R, Hu YC (2008) Context-based routing: technique, applications, and experience. In: Proceedings of NSDI '08, San Francisco
6. Dong J, Curtmola R, Nita-Rotaru C (2009) Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In: Proceedings of WiSec '09, ACM Press, Zurich
7. Dong J, Curtmola R, Nita-Rotaru C (2009) Secure network coding for wireless mesh networks: threats, challenges, and directions. Comput Commun 32(17):1790–1801
8. Eriksson J, Krishnamurthy S, Faloutsos M (2006) Truelink: a practical countermeasure to the wormhole attack in wireless networks. In: Proceedings of ICNP '06, Santa Barbara
9. Gkantsidis C, Rodriguez Rodriguez P (2006) Cooperative security for network coding file distribution. In: Proceedings of INFOCOM 2006, Barcelona
10. Ho T, Leong B, Koetter R, Medard M, Effros M, Karger D (2004) Byzantine modification detection in multicast networks using randomized network coding. In: Proceedings of ISIT '04, Chicago
11. Hu YC, Perrig A, Johnson DB (2003) Packet leashes: a defense against wormhole attacks in wireless adhoc networks. In: INFOCOM '03, San Franciso
12. Jaggi S, Langberg M, Katti S, Ho T, Katabi D, Medard M (2007) Resilient network coding in the presence of byzantine adversaries. In: Proceedings of INFOCOM'07, Anchorage
13. Katti S, Rahul H, Hu W, Katabi D, Médard M, Crowcroft J (2006) Xors in the air: practical wireless network coding. In: Proceedings of ACM SIGCOMM '06, ACM Press, Pisa
14. Kehdi E, Li B (2009) Null keys: limiting malicious attacks via null space properties of network coding. In: Proceedings of IEEE INFOCOM '09, Brazil, pp 1224–1232
15. Krohn M, Freedman M, Mazieres D (2004) On-the-fly verification of rateless erasure codes for efficient content distribution. In: Proceedings of symposium on security and privacy 2004, Oakland
16. Le J, Lui JCS, Chiu DM (2008) DCAR: distributed coding-aware routing in wireless networks. In: Proceedings of ICDCS '08, Beijing
17. Li Q, Chiu DM, Lui J (2006) On the practical and security issues of batch content distribution via network coding. In: Proceedings of ICNP '06, Santa Barbara, California
18. Yu Z, Wei Y, Ramkumar B, Guan Y (2008) An efficient signature-based scheme for securing network coding against pollution attacks. In: Proceedings of INFOCOM 08, San Francisco
19. Yu Z, Wei Y, Ramkumar B, Guan Y (2009) An efficient signature scheme for securing x or network coding against pollution attacks. In: INFOCOM '09, Rio de Janeiro
20. Zhang X, Li B (2008) Dice: a game theoretic framework for wireless multipath network coding. In: Proceedings of Mobihoc 2008, Hong Kong, pp 293–302
21. Zhang X, Li B (2008) Optimized multipath network coding in lossy wireless networks. In: Proceedings of ICDCS '08, Beijing
22. Zhao F, Kalker T, Medard M, Han K (2007) Signatures for content distribution with network coding. In: Proceedings of ISIT '07, Nice, France

Secure Networks Design

►Security of Wireless Mesh Networks (General Overview)

Secure Remote Programming

►Secure Code Dissemination in Wireless Sensor Networks

Secure Routing in Wireless Mesh Networks

Cristina Nita-Rotaru[1], Jing Dong[1], Reza Curtmola[2]
[1]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA

Synonyms

Secure communication

Definition

A secure routing protocol guarantees that the routing service works correctly, i.e., delivers packets between a source and one or multiple destinations without being altered, delayed, or dropped by intermediate forwarder nodes.

Background

Routing protocols ensure that data sent by the source is routed across the network and reaches the destination. The destination can be a single node (unicast routing) or multiple nodes (multicast routing). Most of the routing protocols for ad hoc wireless networks have a route establishment phase where a routing path is established and selected based on a metric (typically hop-count) and a data forwarding phase in which nodes on the path forwards packets to a particular destination.

Wireless mesh networks (WMNs) share a lot of similarities with mobile ad hoc networks (MANETs), as in both, communication is done using wireless transmissions in a multi-hop fashion (i.e., the communication between source and destination is relayed by intermediate nodes). Thus, routing protocols that were proposed in MANETs for unicast (AODV [26], DSR [15], DSDV [25]) and multicast (MAODV [29]) can be used in WMNs. However, some routing protocols such as ODMRP [21] were specifically designed to take advantage of the mesh structure in a WMN. Moreover, given their static nature, WMNs are used primarily for applications that require high throughput. To maximize path throughput, an important focus in routing for WMNs is to use high-throughput metrics (such as ETX [6], PP [9, 18], RTT [2], SPP [29]), in which the paths are selected based on the quality of the wireless links, as opposed to the more traditional hop count metric which seeks to minimize the number of hops on a path. As a result, WMNs share many security threats with MANETs, but they also add new security challenges that originate from the different nature of routing protocols designed to take into account the mesh structure and the high throughput metrics.

Theory

Attacks Against Routing

The main goal of an attack against routing is that of disrupting packet delivery between source and destination (A related attack is when a passive attacker performs traffic analysis, which will not be considered here). One criterion to classify attacks is depending on whether the adversary possesses the credentials to participate in routing. In the absence of any security mechanisms, the wireless transmission medium makes routing vulnerable to outside attacks, in which adversaries replay, inject, or modify control or data packets. These attacks can be prevented by using cryptographic mechanisms such as MACs and digital signatures that ensure authentication and integrity of messages exchanged between nodes. However, such mechanisms are not sufficient against insider attacks, which occur when adversaries gain access to the cryptographic material stored at the nodes. Given the multi-hop nature of routing and the lack of physical security which makes devices susceptible to theft, insider attacks can be a severe security threat.

Another criterion to classify attacks is depending on which phase of the routing protocol is targeted. Attacks can target the path establishment phase and the packet forwarding phase. Both must be secured. Attacks that disrupt the correct path establishment target the underlying routing structure (i.e., a path or multi-paths for unicast, and a mesh or a tree for multicast). Examples are distorting the routing structure or preventing its establishment altogether. The most common attack targeting the packet forwarding phase is the packet dropping attack in which adversaries drop (either partially or totally) the packets that are routed through them. Some attacks target both phases: adversaries manipulate the route establishment phase in order to be included on many routes and subsequently disrupt large portions of the network by attacking the data forwarding phase. Examples include wormhole [13], rushing [14], and metric manipulation attacks [7]. Finally, adversaries can perform the attack by acting individually, or by colluding with other adversaries. The presence of multiple colluding adversaries usually makes an attack more challenging to mitigate.

In a **wormhole attack**, two colluding adversaries create the appearance of a short path by tunneling packets among themselves. For example, in Fig. 1, adversarial nodes $A1$ and $A2$ have created a wormhole link between them (the dotted line), and honest nodes $N1$ and $N2$ choose the route that includes the wormhole link because it seems to have a low cost. In a traditional wormhole attack, $N1$ and $N2$ are outside attackers that simply act as repeaters (e.g., $A1$ sends to $A2$ packets overheard from $N1$, and $A2$ rebroadcast them), leading $N1$ and $N2$ to believe they are direct neighbors. In

Secure Routing in Wireless Mesh Networks. Fig. 1 Wormhole attack

the Byzantine version of the wormhole attack [3], $A1$ and $A2$ are compromised nodes that establish a wormhole link between them, but are still visible on the path between $N1$ and $N2$.

In **rushing attacks**, an attacker is able to propagate an adversarial variant of a packet through the network. A typical example is to take advantage of the flood duplicate suppression technique in the route discovery phase of an on-demand routing protocol such as AODV [26]. Attacker nodes can simply transmit flood packets by ignoring the small randomized delays (required by wireless transmissions in order to reduce collisions). As a result, flood packets traveling through legitimate routes will be ignored and, instead, attackers will be included on routes.

Metric manipulation attacks [7] occur in WMNs that are optimized to use high-throughput metrics (instead of the more traditional hop-count metric). In a typical use, each node is required to collect local information about its adjacent links based on periodical probes from its neighbors. This local information is then propagated in the network, allowing nodes to obtain global information about the quality of routes. In a metric manipulation attack, an adversarial node advertises "high quality" paths by manipulating either its own contribution to the path metric or the contributions of other nodes that were accumulated in the path metric. As a result, attacker nodes will act as "attractors" for many routes. These attacks are Byzantine in nature, as they are conducted by nodes that have the credentials to participate in routing, yet are under adversarial control.

Defenses

Defenses Against Packet Dropping

The *watchdog* technique [21] relies on the fact that a node can overhear its neighboring nodes forwarding packets to other destinations. If a node does not overhear a neighbor forwarding more than a threshold number of packets, it concludes that the neighbor is adversarial. This technique is most effective against individual non-colluding attackers.

Defenses Against Rushing

Rushing Attack Prevention (RAP) [14] prevents the rushing attack by waiting for up to k flood requests and then randomly selecting one to forward, rather than only forwarding the first one. ODSBR [2] adopts another defense against the rushing attack by processing all duplicate flood packets, and if a valid flood packet with a lower metric is received, an additional re-broadcast is scheduled. The advantage of this technique is that even if an adversary performs a successful "rush" in an attempt to be selected on the path, the adversarial variant of the flood will be shortly overridden by the legitimate flood with a lower path cost.

Defenses Against Wormhole

Several defenses were proposed against the traditional wormhole attack. Packet leashes [13] restrict the maximum transmission distance by using either a tight time synchronization (temporal leash) or location information (geographic leash). As such, they require specialized hardware, such as accurate clocks or GPS receivers. Another technique relies on directional antennas [10] and prevents wormholes by having each node maintain accurate information about its neighbors. Messages coming from a node that is not perceived as a neighbor are ignored. TrueLink[9] prevents formation of wormholes by using MAC layer acknowledgments to infer if a link exists or not between two nodes.

The above techniques rely on the end points of the wormhole to prevent wormhole formation. Since the end points of a Byzantine wormhole are adversarial and cannot be trusted to follow the protocol correctly, these techniques are ineffective against Byzantine wormhole attacks. The ODSBR [2] protocol incorporates several techniques to defend against the Byzantine wormhole attack, which are motivated by the observation that the primary wormhole attack is the dropping of packets that attempt to travel through the wormhole, rather than the wormhole formation. A wormhole attack will appear to ODSBR as a faulty link existing between two nodes. ODSBR mitigates the attack not by preventing the formation of the wormhole, but by detecting it and increasing its weight. Once the wormhole's link weight has been increased sufficiently, ODSBR will avoid it and select the next best alternate path.

Defenses Against Metric Manipulation Attacks

Metric manipulation attacks not only distort path establishment, but also make recovery difficult even if attackers are identified, because honest nodes cannot rely on the "poisoned" metric. S-ODMRP [8] uses a measurement-based detection mechanism to first identify attacker nodes and then isolate them through an accusation-based reaction mechanism. The detection component is based on the ability of honest nodes to detect a discrepancy between the advertised data rate and the actual perceived data rate. The accusation component temporarily prevents attacker nodes to participate in routing. To address the metric poisoning effect, the metric is refreshed in the network shortly after attacker identification.

Securing Unicast Routing

Unlike the previous defenses which target individual attacks, several full-fledged secure routing protocols were also proposed to provide resilience against outside [22, 30, 33] and inside [11, 12, 19, 23] attacks.

SRP [24] prevents impersonation and replay attacks for on-demand routing by disabling route caching and providing end-to-end authentication using HMACs [31]. One-way hash chains and digital signatures were used to secure on demand distance vector (SAODV [33], SEAD [11], ARAN [30]) and link state ([24]) routing protocols. SDT [23] and Ariadne [12] use multi-path routing to mitigate data dropping attacks. SDT disseminates packets across several disjoint paths and uses authenticated destination-to-source acknowledgments as proof that packets reached their destination. In Ariadne, nodes that have several paths available assign a fraction of packets to be sent along each path. This multi-path approach has relatively low overhead, converges quickly, and works effectively in a well-connected wireless network, where the number of disjoint paths is large. In Afora [19], nodes do not forward all route replies, but probabilistically drop some of them to increase resilience to attacks. Both Ariadne and Afora rely on an external feedback mechanism to notify them about the resilience of particular paths.

The ODSBR [2] routing protocol was designed to withstand a large set of Byzantine attacks from multiple colluding adversaries, including data dropping, rushing, and wormhole attacks. ODSBR uses an acknowledgment-based feedback technique to detect the presence of attacks, a binary searched-based probing technique to localize faulty links on routes, and selects routes based on a reliability metric that captures link failures and adversarial behavior.

Securing Multicast Routing

Most of the work on multicast routing in WMNs has focused on application specific security issues such as key management, which is needed to ensure data confidentiality and authenticity [3, 4, 16, 18, 32]. In contrast to secure unicast routing, the work studying security issues specific to multicast routing in WMNs is relatively scarce. For tree-based multicast routing protocols, such as MAODV [29], Roy et al. [27] propose an authentication framework which allows the protocol to withstand several outside attacks targeted against the creation and maintenance of the multicast tree. The BSMR [6] protocol complements this work and achieves protection against Byzantine attacks in tree-based multicast protocols by using a combination of techniques which include: Detecting the reliability of links by comparing the perceived data rate with the rate advertised by the multicast source, and selecting routes based on a reliability metric that captures adversarial behavior. Finally, for multicast routing protocols that use a mesh-based routing structure (e.g., ODMRP [20]) in combination with high-throughput metrics, Dong et al. [7] present a defense technique against metric manipulation attacks.

Applications

Secure routing protocols serve as a fundamental building block for any communication system. In the context of wireless mesh networks, applications that can benefit from secure routing include application that need high performance and availability such as multimedia conferencing, and online video broadcasting, as well as applications that require deployment in adversarial environments such as military battle fields and rescue missions.

Recommended Reading

1. Adya A, Bahl P, Padhye J, Wolman A, Zhou L (2004) A multi-radio unification protocol for ieee 802.11 wireless networks. In: Proceedings of BroadNets '04 Lau- sanne, Switzerland, pp 344–354 (2004)
2. Awerbuch B, Curtmola R, Holmer D, Nita-Rotaru C, Rubens H (2008) ODSBR: an on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks. ACM Transactions on Information and System Security (TISSEC), 10(4), January 2008
3. Balachandran RK, Ramamurthy B, Xukai Z, Vinodchandran NV (2005) CRTDH: an efficient key agreement scheme for secure group communications in wireless ad hoc networks. In: Proceedings of ICC 2005
4. Bruschi D, Rosti E (2002) Secure multicast in wireless networks of mobile hosts: protocols and issues. Mob Netw Appl 7(6):503–511
5. Couto DSJD, Aguayo D, Bicket JC, Morris R (2003) A high-throughput path metric for multi-hop wireless routing. In: Proceedings of MOBICOM '03 (2003), ACM, San Diego, pp 134–146
6. Curtmola R, Nita-Rotaru C (2009) BSMR: Byzantine-resilient secure multicast routing in multi-hop wireless networks. IEEE Transactions on Mobile Computing (TMC) 8(4):445–459
7. Dong J, Curtmola R, Nita-Rotaru C (2008) On the pitfalls of using high-throughput multicast metrics in adversarial wireless mesh networks. In: Proceedings of the 4th IEEE conference on sensor, mesh and adhoc communications and Networks (SECON '08) San Francisco
8. Draves R, Padhye J, Zill B (2004) Comparison of routing metrics for static multi-hop wireless networks. In: Proceedings of SIGCOMM '04, ACM Press, New York, pp 133–144
9. Eriksson J, Krishnamurthy SV, Faloutsos M (2006) Truelink: a practical countermeasure to the wormhole attack in wireless networks. In: Proc. of ICNP '2006, Santa Barbara
10. Hu L, Evans D (2004) Using directional antennas to prevent wormhole attacks. In: Proceedings of NDSS, San Diego, 2004
11. Hu YC, Johnson DB, Perrig A (2003) Sead: secure efficient distance vector routing for mobile wireless ad hoc networks. Ad Hoc Networks 1(1):175–192

12. Hu YC, Perrig A, Johnson DB (2002) Ariadne: A secure on-demand routing protocol for ad hoc networks. In: Proceedings of MOBICOM, Atlanta, Georgia, pp 12–23
13. Hu YC, Perrig A, Johnson DB (2003) Packet leashes: a defense against wormhole attacks in wireless ad hoc networks. In: Proceedings of INFOCOM, San Franciso, April 2003
14. Hu YC, Perrig A, Johnson DB (2003) Rushing attacks and defense in wireless ad hoc network routing protocols. In: Proceedings of WiSe, ACM Press, San Diego, pp 30–40
15. Johnson DB, Maltz DA, Broch J (2001) DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins CE (ed) Ad hoc networking, chapter 5. Addison-Wesley, Boston, pp 139–172
16. Kaya T, Lin G, Noubir G, Yilmaz A (2003) Secure multicast groups on ad hoc networks. In: Proceedings of SASN'03 (2003), ACM Press, New York, pp 94–102
17. Keshav S (1993) A control-theoretic approach to flow control. In: Proceedings of the conference on communications architecture and protocols (1993), pp 3–15
18. Lazos L, Poovendran R (2007) Power proximity based key management for secure multicast in ad hoc networks. J Wirel Netw 13(1):127–148
19. Lee HI (2002) Afora: Ad hoc routing in the face of misbehaving nodes. Master's Thesis, MIT, September 2002
20. Lee SJ, Su W, Gerla M (2002) On-demand multicast routing protocol in multihop wireless mobile networks. Mob Netw Appl 7(6):441–453
21. Marti S, Giuli T, Lai K, Baker M (2000) Mitigating routing misbehavior in mobile ad hoc networks. In: Proceedings of MOBICOM, Boston (2000)
22. Papadimitratos P, Haas ZJ (2002) Secure routing for mobile ad hoc networks. In: Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS), San Antonio, pp 27–31
23. Papadimitratos P, Haas ZJ (2003) Secure data transmission in mobile ad hoc networks. In: Proceedings of ACM Workshop on Wireless Security (WiSe), San Diego, 2003
24. Papadimitratos P, Haas ZJ (2003) Secure link state routing for mobile ad hoc networks. In: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops) (2003), IEEE Computer Society, Orlando
25. Perkins CE, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM (1994), London
26. Perkins CE, Royer EM (2000) Ad hoc on-demand distance vector routing. In: Perkins CE (ed) Ad hoc networking. Addison-Wesley, Boston
27. Roy S, Addada VG, Setia S, Jajodia S (2005) Securing MAODV: attacks and countermeasures. In: Proceedings of SECON '05, IEEE 2005
28. Roy S, Koutsonikolas D, Das S, Hu C (2006) High-throughput multicast routing metrics in wireless mesh networks. In: ICDCS 2006, Lisboa
29. Royer E, Perkins C (2000) Multicast ad-hoc on-demand distance vector (MAODV) routing. In: Internet Draft (July 2000)
30. Sanzgiri K, Dahill B, Levine BN, Shields C, Belding-Royer E (2002) A secure routing protocol for ad hoc networks. In: Proceedings of ICNP 2002, Paris
31. The Keyed-Hash Message Authentication Code (HMAC). No. FIPS 198. National Institute for Standards and Technology (NIST), 2002. http://csrc.nist.gov/publications/fips/index.html.
32. Zapata MG, Asokan N (2002) Securing ad hoc routing protocols. In: Proceedings of ACM Workshop on Wireless Security (WiSe) ACM Press, New York, 2002
33. Zhu S, Setia S, Xu S, Jajodia S (2004) Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Proceedings of Mobiquitos (2004), IEEE, pp 42–51

Secure Routing Protocols

Qijun Gu
Department of Computer Science, Texas State University-San Marcos, San Marcos, Texas, USA

Related Concepts

►Authentication

Definition

Secure routing protocols in ad hoc networks are the protocols designed to counteract routing attacks that disrupt route discovery.

Background

In an ad hoc network, nodes form the network collaboratively without using any infrastructure. Nodes do not have the global view on the network topology. In order to deliver packets, they must discover valid routes along which nodes can forward packets from source to destination. The route discovery procedure requires nodes to announce their presence, listen for others' announcements, forward route discovery request and reply packets, maintain and update their routing tables, and share their routing information.

Besides the nodes' mobility and the unreliable wireless communication that already make ad hoc routing protocols difficult, security adds another challenge. As attackers can infiltrate easily in a network and nodes can be compromised, ad hoc routing can be disrupted easily under attacks of forging routing packets, dropping routing packets, detouring routing packets, flushing routine packets, or tunneling routing packets. Such attacks mislead or deceive good nodes to believe that an alive link has failed, a path is disconnected, a short path becomes longer, a long path becomes shorter, or a legitimate and fresh routing packet is redundant. Such attacks essentially change the nodes' views on the network topology and lead to nonoptimal routes and a degradation of network performance. Security routing protocols are created to guard ad hoc networks against these attacks.

Theory

The major security concern aroused by the ad hoc routing attacks is essentially on the integrity of routing protocols. Hence, the fundamental building blocks of secure ad hoc routing protocols include a component for key establishment and management and a component for routing packet authentication. The key component establishes and updates cryptographic credentials among nodes. The credentials are used to build keys among nodes for verifying node identities and packet integrity. The authentication component provides the actual mechanisms to authenticate and verify packets and node identities.

Some ad hoc routing protocols require shared keys between all pairs of nodes participating in route discovery. One approach to distribute shared keys is to load the shared keys into each pair of nodes before deployment. Another approach is to use side channels to exchange and build the shared keys. Other ad hoc routing protocols use public keys. Nodes have a list of trusted public keys. They use key-exchange protocols to compute shared session keys during route discovery. Both key management schemes have issues in incremental network deployment. When new nodes join the network, it is difficult for existing nodes to obtain and verify either shared keys or public keys associated with the new nodes. The distribution of shared keys could be intercepted or eavesdropped by attackers and thus results in leakage of credentials. The distribution of public keys requires trustable authorities, which are hardly practical in ad hoc networks. To address these issues, some key management schemes were proposed on the ideas of binding keys with node addresses, using certificates from a certificate authorities, build transitive trust, and so on. Another challenge of key distribution is revocation. When a node shall be removed from a network, its associated credentials should also be revoked. Such revocated information is flooded throughout the network to ensure all the nodes in the network are aware of the revocation.

The authentication component is designed according to two major types of ad hoc routing protocols: distance vector ad hoc routing and source ad hoc routing. In distance vector routing, each node maintains a routing table of all possible destinations. Each entry in the routing table has the shortest known distance to the corresponding destination. The routing protocol is designed to propagate the distances such that all nodes can keep updating their routing tables upon receiving routing packets to ensure they have the shortest paths to the destinations. In contrast, nodes in source routing do not use a routing table. Instead, they add their addresses in the routing packets that they forward during route discovery. Hence, the routing packets in fact carry the candidate paths. Nodes keep the path information from received routing packets and select the best paths accordingly. Different authentication mechanisms are designed according to the characteristics of the two types of ad hoc routing protocols.

Secure ad hoc on demand distance vector (SAODV) routing protocol [1] is a representative secure distance vector ad hoc routing protocol. In SAODV, a route request packet is designed to carry the request information and a hop count for discovering a route. Routing nodes increment the hop count when forwarding the routing packets. Hence, the authentication in SAODV includes a signature for the invariant fields of routing packets and a hash for the hop count. Given a network of up to m hops, the routing originator generates a one-way hash chain of length equal to the maximum hop $h_0, h_1, h_2, ..., h_m$, so that $h_i = H(h_{i+1})$. The anchor h_0 and the maximum hop count m are put into the invariant field of routing packets. A receiving node can verify the hop count by following the hash chain from the current hop count to the anchor. An example of SAODV is shown in Table 1. In this example, a source S wants to discover a route to a destination D. S broadcasts a route request packet. The packet is then rebroadcast by nodes A, B, and C until it reaches D. When each node rebroadcasts the request packet, it verifies the invariant part of the packet, which is $(REQUEST, id, S, seq_S, D, h_0, m)$ signed by the source's key K_S. Then, it verifies the current hop count i by checking if $h_0 = H^{m-i}(h_{m-i})$. For instance, when node B receives the request, the hop count is 1. Hence, node B checks if $h_0 = H^{m-1}(h_{m-1})$. If all checks pass, the node increments the hop count and also hashes the current hop count hash. When the request packet reaches D, D replies. The reply packet is authenticated and verified in the same way as the request packet. The protocol uses a hash function to ensure that the variant information in routing packets cannot be changed reversely. Hence, although attackers

Secure Routing Protocols. Table 1 An example of route discovery in SAODV

Route request and reply packets	
$S \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 0, h_m >$
$A \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 1, h_{m-1} >$
$B \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 2, h_{m-2} >$
$C \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 3, h_{m-3} >$
$D \rightarrow C$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 0, h'_m >$
$C \rightarrow B$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 1, h'_{m-1} >$
$B \rightarrow A$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 2, h'_{m-2} >$
$A \rightarrow S$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 3, h'_{m-3} >$

cannot make a worse route better, attackers could make a better route worse. Secure efficient ad hoc distance (SEAD) vector routing protocol [2] is another secure distance vector ad hoc routing protocol. Similar to SAODV, it uses a hash function to authenticate the routing metric and the sequence numbers in routing tables. The authentication provides a lower bound on the metric. Although an attacker can increase the metric or claim the same metric, the attacker cannot decrease the metric.

Ariadne [3] is a representative protocol for securing dynamic source routing protocol. When a source node needs to discover a route, it broadcasts a route request packet. Each node, when forwarding the packet, adds itself into the packet as a router on the candidate path. To convince the destination of the legitimacy of each field in the request, each node also adds a message authentication code (MAC) computed over the incremented packet. Hence, as the packet gets closer to the destination, it is incremented with more routers and more MACs. The destination can verify the authenticity of the request packet by checking all MACs. Ariadne assumes a key distribution scheme exists to ensure that the MACs can be computed and verified with the correct and trusted keys. The key distribution in Ariadne is more complicated than SAODV as the MACs require all nodes along the path to have a trustable key to compute the corresponding MAC. An example of Ariadne is shown in Table 2. In the example, a source S wants to discover a route to a destination D, and the path actually goes through nodes A, B, and C. When a node receives the route request packet, it recomputes a per-hop hash in the packet, adds itself to the path and then computes a new MAC. For instance, when B receives the request, it computes $h_2 = H(B, h1)$ and replaces h_1 with h_2. B then adds itself to the path (A, B) and adds a new MAC $M_B = MAC_{K_{B_{ti}}}(Request, S, D, id, ti, h_2, (A, B), (M_A))$. When the destination replies, it adds another MAC $M_D = MAC_{K_D}(Request, S, D, id, ti, h_2, (A, B), (M_A))$ to ensure that the path will not be changed in the reply. Ariadne does not address all security problems in ad hoc routing. For example, an attacking node in the path could remove previous hops that are adjacent to the attacker so that a long path seems shorter.

Secure Routing Protocols. Table 2 An example of route discovery in Ariadne

Route request and reply packets
$S \rightarrow * :< Request, S, D, id, ti, h_0, (), () >$
$A \rightarrow * :< Request, S, D, id, ti, h_1, (A), (M_A) >$
$B \rightarrow * :< Request, S, D, id, ti, h_2, (A, B), (M_A, M_B) >$
$C \rightarrow * :< Request, S, D, id, ti, h_3, (A, B, C), (M_A, M_B, M_C) >$
$D \rightarrow C :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, () >$
$C \rightarrow B :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}) >$
$B \rightarrow A :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, K_{B_{ti}}) >$
$A \rightarrow S :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, K_{B_{ti}}, K_{A_{ti}}) >$

Applications

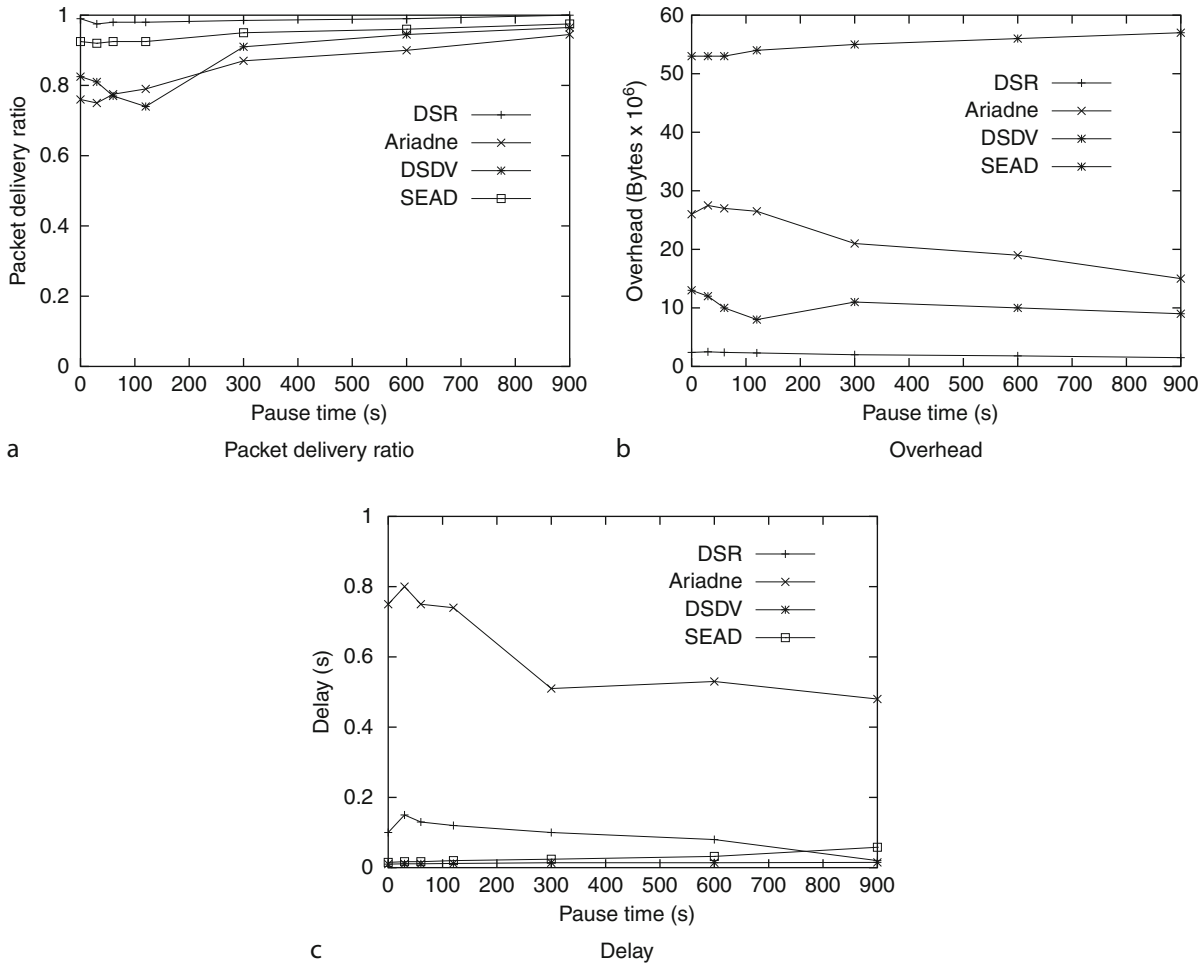
Secure ad hoc routing protocols can be deployed for a variety of applications that have security concerns on their communication. The applications include military operations, emergency disaster responses, community networking, ad hoc meetings, and so on. In these ad hoc networking applications, secure routing protocols can protect the applications against routing attacks.

Experimental Results

Secure ad hoc routing protocols add extra credentials in the original routing packets and require additional procedures for authenticating and verifying routing packets. The security mechanisms thus impact the performance of the secure protocols. Three metrics are used for measuring the performance: packet delivery ratio, latency, and overhead. Packet delivery ratio shows the percentage of delivered packets. Latency shows the time for a packet to reach its destination. Overhead shows the traffic volume of routing packets. Simulated experiments [2, 3] were conducted to compare the performance of secure routing protocols and their nonsecure versions, i.e., Ariadne vs. dynamic source routing (DSR), and SEAD vs. destination sequenced distance vector (DSDV). A network of 50 mobile nodes are simulated. Each node stays at a location and pauses for a period of time called the pause time. Then, it chooses a new location at random and moves there at the maximum speed of 20 m/s.

Figure 1a shows the packet delivery ratio of the four routing protocols. SEAD consistently outperforms DSDV on this metric, because SEAD sends more routing advertisements than DSDV, which allows nodes to obtain more up-to-date routing information. In contrast, Ariadne performs worse than DSR. In route discovery operations, Ariadne requires nodes to wait longer time for keys to be disclosed. Then, routing packets can be verified and routes can be established. Hence, packets are more likely to time out, as routes have a shorter life time.

Figure 1b shows the overhead of the four routing protocols. Both SEAD and Ariadne have larger overhead than their nonsecure versions, because secure routing packets are larger in size in order to carry more authentication



Secure Routing Protocols. Fig. 1 Performance of secure routing protocols, reported in [2, 3]

information, such as the hash value in routing advertisement in SEAD and the authentication tokens in all routing packets in Ariadne. In addition, SEAD also increases the number of routing advertisements, which results in more routing packets.

Figure 1c shows the latency of the four routing protocols. SEAD has higher latency than DSDV, because SEAD needs more network capacity to deliver more larger routing packets. Hence, normal data packet transmission is generally delayed due to the reduced network capacity. The latency caused by transmitting extra security information in routing packets is more obvious in the comparison of Ariadne and DSR. DSR itself uses routing packets larger than DSDV and SEAD. Ariadne further adds authentication tokens into routing packets. Hence, using DSR results in larger latency than DSDV and SEAD, while using Ariadne results in even larger latency than DSR.

Open Problems

Secure ad hoc routing protocols are built upon secure key distribution and management, as the keys are needed in authenticating and verifying routing packets. However, having such a key management in ad hoc networks is hard due to the mobile and infrastructure-less nature of ad hoc networks. It needs more research to find good key managements for routing protocols. Furthermore, many attacks against ad hoc routing protocols have been identified. The current secure routing protocols can address a portion of the attacks, while leaving networks unprotected from the other attacks. New secure routing protocols are still needed to address those attacks.

Recommended Reading

1. Zapata MG, Asokan N (2002) Securing ad hoc routing protocols. Proceedings of ACM Workshop on Wireless Security, pp 1–10

2. Hu Y-C, Johnson DB, Perrig A (2003) SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. Ad Hoc Networks 1:175–192
3. Hu Y-C, Perrig A, Johnson DB (2005) Ariadne: a secure on-demand routing protocol for ad hoc networks. Wireless Networks 11(1–2):21–38

Secure Shell

►SSH

Secure Signatures from the "Strong RSA" Assumption

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Cramer–Shoup Public Key Scheme; ►Digital Signature Schemes

Background

In the late 1990s it was realized that by making a somewhat stronger intractability assumption than RSA (►RSA Problem), it is possible to devise ►digital signature schemes that are fairly efficient, and at the same time have a rigorous proof of security (without resorting to the ►random-oracle heuristic). The intractability assumption states that given a modulus n (►modular arithmetic) of unknown factorization and an element x in the ►ring Z_n^*, it is hard to come up with an exponent $e \geq 2$ and an element y in Z_n^*, such that $y^e = x \pmod{n}$. This assumption, first used by Barić and Pfitzmann in the context of ►fail-stop signatures [1], is called the *strong RSA assumption* (or the *flexible RSA assumption*).

Theory

A simple way of using this assumption for signatures was described by Gennaro et al. [7]. In their construction, the public key (►public key cryptography) is a triple (n, x, H), where n is product of two "quasi-safe primes," x is a random element in Z_n^*, and H is a ►hash function, mapping strings to odd integers. The secret key consists of the factorization of n. To sign a message m, the signer picks a random string r, computes $e \leftarrow H(m, r)$, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x \pmod{n}$. To verify a signature (r, y) on message m, the verifier checks that $y^{H(m,r)} = x \pmod{n}$.

Gennaro et al. proved that this scheme is secure – in the sense of existential unforgeability (►existential forgery) under an adaptive chosen message attack (EU-CMA) – under some conditions on the function H. The first condition is that H is *division intractable*. This means that it is hard to come up with a list of pairs, (m_i, r_i), $i = 1, \ldots, t$, where the integer $H(m_t, r_t)$ divides the product $\prod_{i=1}^{t=1} H(m_i, r)$. The other condition on H means, informally, that one cannot reduce breaking the strong RSA assumption to "breaking the ►hash function H." It is shown in [7] that hash functions satisfying these conditions exist if the strong RSA assumption holds. (It is also shown in [7] that if H is modeled as a random oracle, then it satisfies these conditions, provided that its output is sufficiently long. However, Coron and Naccache showed in [2] that the output of H in this case should be at least as long as the modulus n.)

Cramer and Shoup [4], followed by Fischlin [6], proposed more efficient signatures based on the strong RSA assumption. In the ►Cramer–Shoup public key scheme, the public key is a 5-tuple (n, h, x, e', H), where n is a product of two "safe primes," h, x are random ►quadratic residues in Z_n^*, e' is an odd prime, and H is a hash function, mapping strings to integers. If ℓ denotes the output length of H, then the prime e' must be at least $(\ell+1)$-bit long. The secret key consists of the factorization of n. To sign a message m, the signer picks a new $(\ell+1)$-bit prime $e \neq e'$ and a random quadratic residue $y \in Z_n^*$, sets $x' \leftarrow (y')^e h^{-H(m)} \pmod{n}$, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = xh^{H(x')} \pmod{n}$. To verify a signature (e, y, y') on message m, the verifier checks that e is an odd $(\ell+1)$-bit number, $e \neq e'$, sets $x' \leftarrow (y')^e h^{-H(m)} \pmod{n}$, and checks that $y^e = xh^{H(x')} \pmod{n}$.

In the scheme of Fischlin, the public key is a 5-tuple (n, g, h, x, H), where n, h, x, H are as in the Cramer–Shoup scheme, and g is yet another random quadratic residue in Z_n^*. Again, the secret key is the factorization of n, and we use ℓ to denote the output length of H. To sign a message m, the signer picks a new $(\ell+1)$-bit prime e and a random ℓ-bit string α, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = xg^{\alpha} h^{\alpha \oplus H(m)} \pmod{n}$. To verify a signature (e, α, y) on message m, the verifier checks that e is an odd $(\ell+1)$-bit number, that α is an ℓ-bit string, and that $y^e = xg^{\alpha} h^{\alpha \oplus H(m)} \pmod{n}$. Fischlin also proposed other variations of this scheme, where the prime e can be chosen even shorter than $\ell+1$ bits (and the computation made slightly more efficient), at the price of a longer public key.

For all of these schemes, it is proved that they are secure (in the sense of EU-CMA), assuming the strong RSA assumption and the collision intractability of the hash function H, and as long as the signer never uses the same prime e for two different signatures. The Cramer–Shoup signature scheme was generalized by Damgård and Koprowski to any ►group where extracting roots is hard [5]. The same generalization can be applied to the schemes described by Fischlin.

It is interesting to note that the Cramer–Shoup signature scheme can be viewed as a simple variation on an earlier scheme by Cramer and Damgård [3]. The Cramer–Damgård scheme is based on a tree construction, and one of its parameters is the depth of that tree. For a tree of depth one, the scheme maintains in the public key a list of t odd primes $e_1, \ldots, e_t$, and can be used to generate up to t signatures, using a different prime each time. The Cramer–Shoup scheme is obtained essentially by letting the signer choose the odd primes "on the fly" instead of committing to them ahead of time in the public key. One pays for this flexibility by having to make a stronger intractability assumption. Since the attacker can now choose the exponent e relative to which it issues the forgery, one has to assume the strong RSA assumption (whereas the standard RSA assumption suffices for the Cramer–Damgård scheme.)

A curious feature of the Cramer–Shoup and Fischlin schemes (as well as some instances of the Gennaro–Halevi–Rabin scheme) is that when there is an a priori polynomial bound on the total number of signatures to be made, the signer can generate its public/private key pair even without knowing the factorization of the modulus n. (This is done using the same strategy as the simulator in the security proof of these schemes.) That makes it possible in some cases to have many users in a system, all sharing the same modulus n.

Recommended Reading

1. Barić N, Pfitzmann B (1997) Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer-Verlag, Berlin, pp 480–494
2. Coron JS, Naccache D (2000) Security analysis of the Gennaro–Halevi–Rabin signature scheme. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer-Verlag, Berlin, pp 91–101
3. Cramer R, Damgård IB (1996) New generations of secure and practical RSA-based signatures. In: Koblitz N (ed) Advances in Cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer-Verlag, Berlin, pp 173–186
4. Cramer R, Shoup V (2000) Signature schemes based on the strong RSA assumption. ACM Trans Inform System Sec (ACM-TISSEC) 3(3):161–185
5. Damgård IB, Koprowski M (2002) Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2002, Lecture notes in computer science, vol 2332. Springer-Verlag, Berlin, pp 256–271
6. Fischlin M (2003) The Cramer-Shoup strong-RSA signature scheme revisited. In: Desmedt YG (ed) Public key cryptography – PKC 2003. Lecture notes in computer science, vol 2567. Springer-Verlag, Berlin, pp 116–129
7. Gennaro R, Halevi S, Rabin T (1999) Secure hash-and-sign signatures without the random oracle. In: Stem J (ed) Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer-Verlag, Berlin, pp 123–139

Secure Socket Layer (SSL)

Clemens Heinrich
Francotyp-Postalia GmbH, Birkenwerder, Germany

Synonyms

SSL; TLS

Related Concepts

►Cryptographic Protocol; ►HTTPS; ►Security Standards Activities; ►Web Security

Definition

Secure Socket Layer (SSL) denotes the predominant communication security protocol of the Internet particularly for World Wide Web (WWW) services relating to electronic commerce or home banking.

Background

The majority of web servers and browsers support SSL as the de facto standard for secure client-server communication. The Secure Socket Layer protocol builds up point-to-point connections that allow private and unimpaired message exchange between strongly authenticated parties.

In the ISO/OSI reference model [8], SSL resides in the session layer between the transport layer (4) and the application layer (7); with respect to the Internet family of protocols, this corresponds to the range between TCP/IP and application protocols such as HTTP, FTP, Telnet, etc. SSL provides no intrinsic synchronization mechanism; it relies on the data link layer below.

History

Netscape developed the first specification of SSL in 1994, but only publicly released and deployed the next version, SSLv2 [6]. With respect to ►public key cryptography, it relies mainly on RSA encryption (►RSA Public

Key Cryptosystem) and ►X.509-compliant ►certificates. ►Block ciphers, such as DES (►Data Encryption Standard), ►Triple DES (3DES), and ►RC4, along with hash functions like ►MD5 and ►SHA, complement the suite of prevailing algorithms.

SSLv3 followed in 1996, adding cryptographic methods such as ►Diffie–Hellman key agreement (DH), support for the FORTEZZA key token, and the ►Digital Signature Standard (DSS) scheme [5].

This article focuses on SSL version 3.0 and its designated successor protocol ►Transport Layer Security (TLS) 1.x, which the Internet Engineering Task Force (IETF) published for the first time in January 1999 [4]. The most recent standard proposal RFC 5246 for TLS 1.2 was published in August 2008 [3].

Theory

Layer Structure

SSL splits into distinct layers and message types (see Fig. 1). The *handshake* message sequence initiates the communication, establishes a set of mutual parameters like the protocol version, applicable cryptographic algorithms (*cipher suites*), and assures the validity of the message sequence. During the handshake, the participants accomplish the negotiated ►authentication and derive the session key material.

The *record layer* fragments the full data stream into records with a maximum size of 2^{14} bytes and envelopes them cryptographically under the current session keys. Records contain a keyed message authentication code (►HMAC). The initial handshake presupposes a NULL cipher suite applying no encryption and no HMAC.

The record layer fully provides the use of compression. However, for patent reasons, the core specifications name no method explicitly, except for the mandatory NULL algorithm, which practically makes compression an incompatible, implementation-dependent feature.

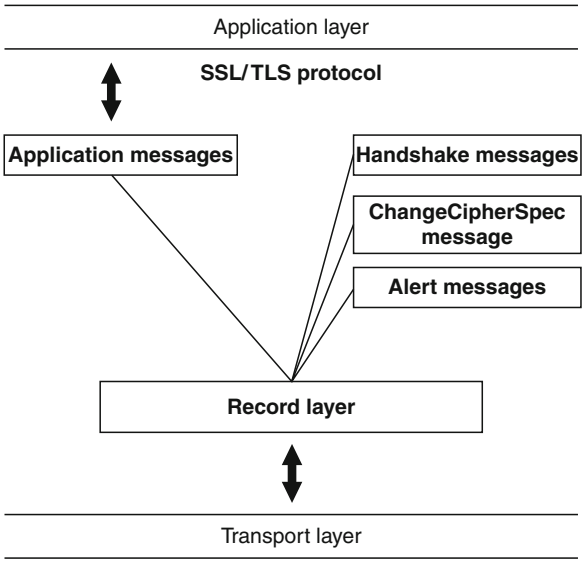
Additional *alert* messages inform on exceptional protocol conditions or one participant's demand to end the communication (*closure* alert).

Basic Protocol Sequence

The SSL handshake accomplishes three goals. Firstly, both parties agree on a *cipher suite*, i.e., the set of cryptographic algorithms that they permit to use for the application data protection.

Secondly, they establish a common *master_secret* in order to derive their session key material.

Finally, the participant's identities are authenticated. Although the SSL specification permits anonymous, server-only, and mutual authentication, it is customary to only assert the server's identity.



Secure Socket Layer (SSL). Fig. 1 SSL layer and message structure

Figure 2 gives an overview of the SSL protocol variants. It comprises four different handshake sequences, each identified by a capital letter:

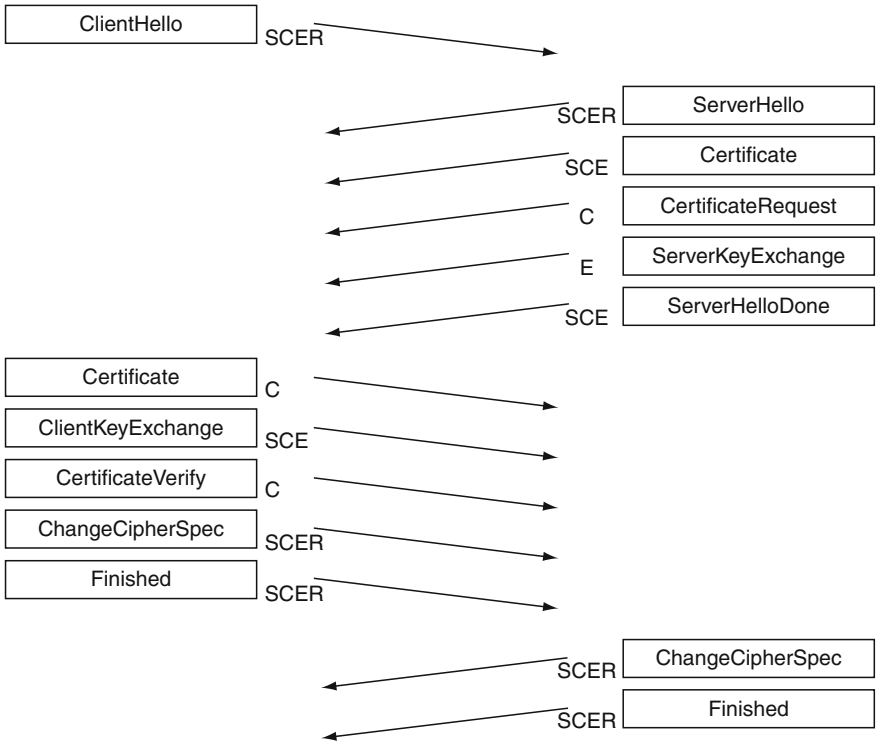
- ***S*** denotes the server-authenticated message flow.
- ***C*** marks the sequence with additional client authentication.
- ***E*** shows the handshake variant with ephemeral ►Diffie–Hellman key agreement.
- ***R*** stands for the handshake of resumed sessions. Note, that the message pairs *ChangeCipherSpec/Finished* of client and server are drawn in reverse order; the server message pair follows *ServerHello* immediately.

The client opens the connection with a *ClientHello* message, which contains its selection of acceptable cipher suites and a random number.

The server chooses from the supported cipher suite and adds another random number, which together builds the *ServerHello* response. Later, these two random numbers become part of the session's *master_secret*.

Server Authentication

The server appends a *Certificate* message, which holds a ►X.509 certificate bearing its identity and public key. Most often RSA keys (►RSA Digital Signature Scheme) are used. DSS-signed certificates usually carry a long-term DH public key. If multiple levels of the public key hierarchy



Secure Socket Layer (SSL). Fig. 2 SSL protocol sequence

separate the server certificate from the root authority certificate present in the client browser, then the server is required to deliver an ordered list of all dependent certificates. The empty *ServerHelloDone* message finishes this sequence.

The client confirms the validity of the certificate chain up to one of its trusted (built-in) root certificates. It generates another random number and encrypts it under the server's RSA public key. This encrypted *pre_master_secret* forms the *ClientKeyExchange* message.

DH/DSS cipher suites might demand the client to create (ephemeral) DH keys matching the server's domain parameters for the *ClientKeyExchange* message. Note, that if both parties own certificates with group-compatible, fixed DH public keys, every connection generates the identical *pre_master_secret*.

Both sides derive the shared session key material independently in two steps. First, the *key derivation function* (KDF) transforms the client's *pre_master_secret* and both exchanged random numbers into the *master_secret*. Afterwards, the KDF is reapplied to the *master_secret* and both random values to compute the final key block. With respect to the chosen cipher suite, the key block is broken up into a maximum of six segments used as directional encryption keys (with initialization vectors) and directional ►HMAC keys.

Client Authentication

SSL servers can demand client authentication through a *CertificateRequest* message. It contains the permitted certificate types, i.e., signature algorithms, and a list of trusted ►certification authorities identified by their respective distinguished name.

The client answers with a *Certificate* message holding either a single certificate or the full certification chain. In addition, it creates a *CertificateVerify* message that contains a digest of the previous handshake messages, signed by the private key that corresponds to the just conveyed certificate.

Ephemeral Diffie–Hellman Key Agreement

The ephemeral ►Diffie–Hellman key agreement (DH) embeds into the *ServerKeyExchange* and the *ClientKeyExchange* messages. Both sides send their DH public keys and, together with their own DH private keys, calculate a shared *pre_master_secret*. Note, that anonymous DH cipher suites are susceptible to man-in-the-middle

attacks and protect only against passive eavesdropping (►eavesdropper).

Both sides end the handshake sequence with two further messages: *ChangeCipherSpec* indicates the shift to the newly negotiated cipher parameters. *Finished* is the first message encrypted under the new cryptographic material; it declares the handshake sequence complete. It holds an HMAC digest over the whole handshake sequence and the negotiated *master_secret* in order to ensure that no message tampering remains undetected.

The cryptographic state is established and confirmed on both sides, and the record layer now encrypts and authenticates application data under its new session keys.

The alert message *CloseAlert* indicates the protocol end, followed by the TCP layer's closing FIN packet.

Protocol Resumption

SSL permits the resumption of formerly established sessions in order to shortcut the handshake and preserve CPU cycles by avoiding, for instance, the computationally expensive *pre_master_secret* decryption.

Depending on whether the server supports this feature, it sends a session identification string enclosed in the *ServerHello* message. After establishing a valid cryptographic state, this id refers to the stored *master_secret*. Subsequent client connections indicate their intention to resume a session by specifying the id in the *ClientHello* message.

Resumed sessions possess unique key blocks because the key generation process recombines the stored *master_secret* with fresh random nonce out of both *Hello* messages.

Additional Information

SSL permits the renegotiation of its cipher suites during the course of the application protocol through a simple repetition of the handshake sequence.

The Internet Assigned Numbers Authority (IANA) allots unique TCP port numbers to SSL/TLS-enabled protocols, which are marked with the appended letter *S*; for example, port 443 for HTTPS or 989/990 for FTPS [7].

Security Analysis

Several authors have analysed the SSL protocol suite, stating in consensus that, beginning with v3.0, it is mature and without major design flaws [11, 12, 14].

Wagner and Schneier conclude in their analysis that "*In general SSL 3.0 provides excellent security against eavesdropping and other passive attacks. Although export-weakened modes offer only minimal confidentiality protection, there is nothing SSL can do to improve that fact.*" [14]

Some minor attacks are known; however, cautious implementation seems to prevent their applicability [13].

The ►man-in-the-middle *version rollback* attack tries to establish a SSL v2.0 handshake protocol between SSLv3/TLS-capable parties in compatibility mode. Due to a serious flaw in SSLv2, an active adversary is capable to enforce an export weakened cipher suite, and attack the session keys *brute-force* (►Cryptanalysis). This SSLv2 attack is also called *cipher suite rollback*. The TLS standard [3] gives recommendations on how to detect downgrade attempts by embedding a short, well-defined pattern into the PKCS#1 padding data (►PKCS) of the RSA encryption, instead of using of purely random bytes. If an SSLv3/TLS-capable server finds the pattern, it will recognize that the other party operates in backwards compatibility mode, although a later protocol version is supported.

Bleichenbacher published an attack against the PKCS#1 (version 1) formatted RSA encryption known as the *million message attack* [1]. Probing an SSL/TLS server with chosen *ClientKeyExchange* messages might reveal the encrypted *pre_master_secret* after about 2^{20} attempts if the server issues error alerts that allow distinguishing between correctly and incorrectly formatted messages (►chosen ciphertext attack). The TLS specification recommends as a countermeasure to treat PKCS#1 formatting errors by simply continuing the handshake protocol with a randomly generated *pre_master_secret*. In this case, the server behaves alike, whether or not the formatting is correct [4].

The general method of timing cryptanalysis [10] is applicable against SSL/TLS servers if a large number of unbiased measurements of private key operations is available. Practical attacks were shown for example by Brumley and Boneh [2] and Klima et al. [9]. Countermeasures against timing attacks usually degrade performance, for instance, by randomly delaying cryptographic operations or by retaining a constant response time in order to conceal information of the algorithms' execution paths.

Recommended Reading

1. Bleichenbacher D (1998) Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS#1. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12
2. Brumley D, Boneh D (2003) Remote timing attacks are practical. In: Proceedings of the 12th USENIX security symposium, Washington, DC
3. Dierks T, Rescorla E (2008) The transport layer security (TLS) protocol version 1.2. IETF RFC 5246. This document obsoletes predecessors RFC 3268, 4346, 4366 and updates RFC 4492. http://www.ietf.org/rfc/rfc5246.txt. Accessed Aug 2008
4. Dierks T, Allen C (1999) The TLS protocol version 1.0. IETF RFC 2246. This document was obsoleted by RFC 4346. http://www.ietf.org/rfc/rfc2246.txt. Accessed Jan 1999

5. Freier A, Karlton P, Kocher P (1996) The SSL 3.0 protocol. Netscape Communications Corp, 1996
6. Hickman KEB (1995) The SSL protocol v2.0 (revised). Netscape Communications Corp., Feb 1995. Note: Precursor documents of v2.0 were published in 1994
7. Internet Assigned Numbers Authority (IANA). http://www.iana.org/
8. ISO 7498-2. Information processing systems – open systems interconnection – basic reference model – part 2: security architecture. ISO International Standard 7498-2; First edition 1989-02-15
9. Klima V, Pokorny O, Rosa T (2003) Attacking RSA-based sessions in SSL/TLS. http://eprint.iacr.org/2003/053/
10. Kocher P (1997) Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: Kaliski Jr BS (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 104–113
11. Mitchell JC, Shmatikov V, Stern U (1998) Finite state analysis of SSL 3.0. In: Proceedings of the 7th USENIX security symposium, San Antonio, TX
12. Paulson LC (1999) Inductive analysis of the Internet protocol TLS. In: Christianson B, Crispo B, Harbison WS, Roe M (eds) Security protocols: 6th international workshop 1998. Lecture notes in computer science, vol 1550. Springer, Berlin, p 13ff
13. Rescorla E (2000) SSL and TLS: designing and building secure systems. Addison-Wesley, Reading
14. Wagner D, Schneier B (1997) Analysis of the SSL 3.0 protocol (revised). In: The second USENIX workshop on electronic commerce proceedings, USENIX Press, Oakland, Nov 1996

Secure Time Synchronization

Kun Sun[1], Peng Ning[2]
[1]Intelligent Automation, Inc., Rockville, MD, USA
[2]Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Related Concepts

►Security; ►Time Synchronization; ►Wireless Sensor Networks

Definition

Secure time synchronization in wireless sensor networks is to securely synchronize the clocks of sensor nodes in hostile environments.

Background

Time synchronization is indispensable for many wireless sensor network applications. Several time synchronization protocols (e.g., [1, 2]) have been proposed for wireless sensor networks in benign environments. However, without addressing security, all those time synchronization techniques cannot survive malicious attacks in hostile environments. Attacks against existing time synchronization protocols have been summarized in [3]. Because all time synchronization protocols rely on time-sensitive message exchanges, an adversary may forge or delay time synchronization messages, jam the communication channel, and drop time synchronization messages. Though message authentication can be used to validate message sources and contents, it cannot validate the timeliness of messages.

Theory

A number of secure time synchronization techniques have been proposed for wireless sensor networks to achieve pair-wise time synchronization between two neighbor nodes, cluster-based time synchronization among a group of sensor nodes, or global time synchronization in a whole sensor network.

First, Ganeriwal et al. [6] proposed a secure single-hop pair-wise synchronization technique to deal with the pulse-delay and wormhole attacks. It uses authenticated time-stamp information to estimate both clock difference and message transmission delay, which can detect the extra message transmission delay introduced by the pulse-delay and wormhole attacks. However, because it uses software to authenticate time synchronization messages, it does not work on sensor radios with high data rate (e.g., 250 kbps) due to the large delay for generating message authentication code. To solve this problem, Sun et al. [4] proposed to use hardware security component in IEEE 802.15.4 compliant radios to authenticate the time synchronization messages. Song et al. [5] proposed two methods for detecting and tolerating the pulse-delay attacks: one transforms attack detection into statistical outliers detection, and the other detects attacks by deriving the bound of the time difference between two nodes through message exchanges.

Second, a fault-tolerant cluster-wise clock synchronization [8] has been proposed to provide a common clock among a cluster of nodes, where the nodes in each cluster can communicate with each other directly through broadcast. It guarantees an upper bound of time difference between normal nodes in a cluster, provided that the malicious nodes are no more than one-third of the cluster. Unlike the traditional fault-tolerant time synchronization approaches, this scheme does not introduce collisions between synchronization messages, nor does it require costly digital signatures.

Third, two secure and resilient global time synchronization schemes [7] have been proposed to achieve a common clock among all the sensor nodes in a sensor network. The basic idea is to provide redundant ways for each node to synchronize its clock with the common source, so that it can tolerate partially missing or false

synchronization information provided by the malicious nodes. A secure and resilient global time synchronization protocol called TinySeRSync [5] has been implemented on MICAz motes running TinyOS. The experimental results indicate that TinySeRSync is a practical system. A secure global time synchronization scheme [9] has been proposed for heterogeneous sensor networks (HSN), which consists of a small number of powerful high-end sensors and a large number of low-end sensors. The scheme consists of two parts: synchronizing all high-end sensors that have long transmission range in an HSN and synchronizing low-end sensors to the high-end sensors in each cluster.

Applications

Time synchronization is an important component of a wireless sensor network to provide a common clock among sensor nodes. Most wireless sensor network applications, such as data fusion, target tracking, and power saving, require synchronized clocks among sensor nodes. The time-slotted MAC protocols achieve multiple accesses to the shared communication medium by assigning time slots to a group of nodes. Thus, sensor nodes need to have a synchronized clock to access their time slots without colliding with each other. However, without addressing security, all the time synchronization techniques in wireless sensor networks cannot survive the malicious attacks in hostile environments. Secure time synchronization can provide a common clock in sensor nodes even under attacks in hostile environments.

Open Problems

WirelessHART [10] is an industrial standard on using wireless sensor networks in industrial control systems. Time synchronization is critical for time division multiple access (TDMA) and channel-hopping techniques used in WirelessHART MAC layer protocol. It is important to measure the impacts of various attacks on the time synchronization scheme in WirelessHART standard and provide countermeasures for those potential attacks.

Recommended Reading

1. Elson J, Girod L, Estrin D (2002) Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Oper Syst Rev 36:147–163
2. Ganeriwal S, Kumar R, Srivastava MB (2003) Timing-sync protocol for sensor networks. In: Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys), Los Angeles, California, pp 138–149
3. Manzo M, Roosta T, Sastry S (2005) Time synchronization attacks in sensor networks. In: Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, Alexandria, Virginia, pp 107–116
4. Sun K, Ning P, Wang C, Liu A, Zhou Y (2006) Tinysersync: secure and resilient time synchronization in wireless sensor networks. In: Proceedings of 13th ACM Conference on Computer and Communications Security (CCS'06), Alexandria, Virginia, pp 42–51
5. Song H, Zhu S, Cao G (2005) Attack-resilient time synchronization for wireless sensor networks. In: Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05), Washington, DC
6. Ganeriwal S, Capkun S, Han C, Srivastava MB (2005) Secure time synchronization service for sensor networks. In: Proceedings of 2005 ACM Workshop on Wireless Security (WiSe 2005), September 2005, Cologne, Germany, pp 97–106
7. Sun K, Ning P, Wang C (2006) Secure and resilient clock synchronization in wireless sensor networks. IEEE J Select Areas Commun 24(2):395–408
8. Sun K, Ning P, Wang C (2005) Fault-tolerant cluster-wise clock synchronization for wireless sensor networks. IEEE Trans Depend Secure Comput 2(3):177–189
9. Du X, Xiao Y, Guizani M, Chen HH (2008) Secure and efficient time synchronization in heterogeneous sensor networks. IEEE Trans Vehicular Technol
10. WirelessHART, http://www.hartcomm.org/protocol/wihart/wireless_technology.html

Secure Vehicular Communication Systems

Panos Papadimitratos, Jean-Pierre Hubaux
School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Definition

Security for vehicular communication (VC) systems comprises architectures and specific schemes to manage faulty behavior of the VC system entities, which are computing, sensing, and communication devices integrated in vehicles and the roadside.

Background

After the deployment of automated toll collection or active roadsigns, VC systems are envisioned as the future technology for improved traffic efficiency and safety. VC systems will comprise *nodes*, in other words, vehicle-mounted and roadside infrastructure units (RSUs), equipped with onboard sensory, processing, and wireless communication modules. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication can

enhance transportation safety and efficiency, as well as infotainment. For example, VC systems can send warnings about environmental hazards (e.g., ice on the pavement), traffic and road conditions (e.g., emergency braking, congestion, or construction sites), and local (e.g., tourist) information.

The unique features of VC are a double-edged sword: a rich set of tools will be available, but a formidable set of abuses and attacks will then become possible. Consider, for example, an attacker that "contaminates" large portions of the vehicular network with false information: A single compromised vehicle can transmit false hazard warnings that can then be taken up by all vehicles in both traffic streams; or a tampered vehicle can forge messages to masquerade as an emergency vehicle to mislead other vehicles to slow down and yield; or a different type of attacker can deploy a number of receivers and record messages transmitted by the vehicles, especially safety beacons that report the vehicle's location, in order to track a vehicle's location and transactions and to infer private information about its driver and passengers.

It is clear that to thwart such attacks, security and privacy-enhancing mechanisms are necessary; in fact, they are a prerequisite for deployment. Otherwise VC systems could make antisocial and criminal behavior easier, in ways that would actually jeopardize the benefits of their deployment. This has been recently well understood in academia, the industry, and among authorities; and a number of concerted efforts have been undertaken to design security architectures for VC systems [1–4].

Adversary Model

VC system entities could deviate from the implemented protocol definitions. Such *faulty* or *adversarial* behavior could be the result of rogue versions of the protocols built by attackers, and modifications of the functionality of the VC system nodes; by tampering either with the onboard software and/or the hardware, or even because of VC software corruption by malicious software (viruses, trojans, etc., usually termed as malware). In general, many adversarial nodes can be present, often acting individually; or possibly in collusion, coordinating their actions. It is reasonable to assume that at any time and location only a few adversaries are likely to be physically present. This does not preclude a group of adversarial nodes surrounding a correct one, but such a situation would be rare. Of course, adversarial behavior can be passive: The attackers deploy or control a possibly large number of devices that collect VC messages and thus information on vehicles and system users.

Security Requirements

Without considering specific applications and protocols, a list of general security requirements are identified. *Message authentication and integrity* to protect messages from alteration and allow receivers to corroborate the node that created the message. If necessary, *entity authentication* can provide evidence of the sender *liveness*, that is, the fact that it generated a message recently. To prevent a sender from denying having sent a message, *non-repudiation* is needed. Furthermore, *access control* and *authorization* can determine what each node is allowed to do in the network, in terms of the implemented system functionality. *Confidentiality* can keep message content secret from unauthorized nodes.

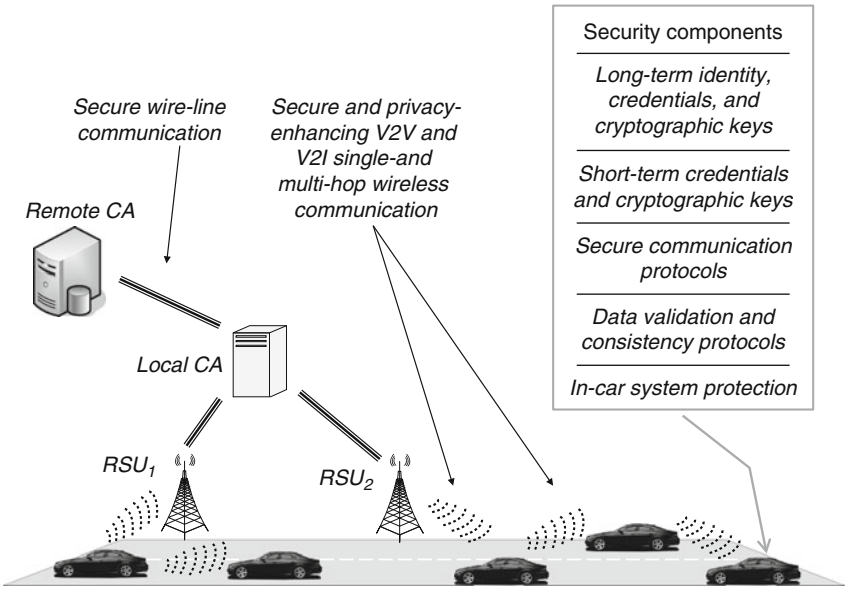
Related to information hiding, *privacy and anonymity* are required, at least at the level of protection achieved before the advent of VC systems. In general, VC systems should not disclose or allow inferences on private user information. The identity of a vehicle performing a VC-specific action (e.g., transmitting a message) should be concealed, and an observer should be unable to determine which vehicle performed an action and should be unable to link any two actions by the same vehicle. But transportation safety applications need to correlate locations of a vehicle over a short period (e.g., to warn about a collision). Thus, a less stringent requirement is considered: messages produced by a node over a protocol-selectable period of time τ can be linked, but any two messages generated at times t_1 and t_2, such that $t_2 > t_1 + \tau$ cannot. The shorter τ is, the fewer the linkable messages are, and the harder it becomes to trace a node.

Along with privacy-enhancing technologies, it is required that any VC-specific action can be linked to the specific related node in order to attribute *liability*, for example, in the case of an accident. Finally, *availability* is also sought, so that VC systems can remain operational even in the presence of faults and can resume normal operation after the removal of the faulty nodes. Another significant dimension is that of *non-cryptographic security*, including the determination of data *correctness* or *consistency*.

Theory and Applications

Security Architecture for VC Systems: Basic Elements

Secure VC systems, illustrated in Fig. 1, will rely on multiple *certification authorities* (CAs), each managing the *long-term* identities and credentials for nodes registered in its *region* (e.g., canton or state). Each node is uniquely identified, and it holds one or more private–public key pairs



Secure Vehicular Communication Systems. Fig. 1 Abstract view of the system

and certificates. The CA attests to the attributes of each registered node, according to its capabilities and roles in the system. The CAs are responsible for *evicting* nodes from the system, for administrative or technical reasons, if needed. The CAs interact infrequently with nodes, utilizing RSUs as a gateway.

The basic technique for nodes to *secure communication* is to digitally sign messages, after attaching a time stamp and the signer's location and certificate to the message. This way, modification, forgery, replay, and relay attacks can be defeated. The relay attacks relate to secure neighbor discovery, which is possible precisely because safety beacons can include the time and location at the point they are sent across the wireless medium. The originator or relaying nodes can sign beacons, multi-hop flooded, and position-based multi- or uni-casted messages in different ways.

To provide both security and a degree of anonymity, long-term keys and credentials are *not* used to secure communication. Rather, the approach of *pseudonymity* or *pseudonymous authentication* is used. Each vehicle is equipped with multiple certified public keys (pseudonyms) that do not reveal the node identity. It obtains those pseudonyms from a trusted third party, a *pseudonym provider*, by proving it is registered with a CA. Then, the vehicle uses each pseudonym and private key for at most τ seconds, the pseudonym lifetime, and then discards it. Messages signed under the same pseudonym can be trivially linked, but messages signed under different pseudonyms cannot.

Security Architecture for VC Systems: Additional Elements

Hardware security: A trusted computing base, termed the hardware security module (HSM) in the SeVeCom architecture [1, 2], has tamper-resistant properties, and it consists of a CPU, nonvolatile memory, a built-in clock, an I/O interface, and a built-in battery. Its main functions are for private key cryptographic operations and provision of trustworthy time stamping.

Revocation: Certificates of faulty nodes or compromised cryptographic keys have to be revoked, to prevent damage to the VC system. Revocation can be decided by the CA for administrative or technical reasons. A distributed protocol can be locally executed to collect evidence of misbehavior, identify wrongdoers, and then exclude them from the local VC operation. The fundamental revocation mechanism is the distribution of *revocation lists* (*RLs*). Sparse roadside infrastructure, with RSUs placed kilometers apart and transmitting the RL at a few kilobits per second, suffices for all vehicles to obtain an RL of hundreds of kilobytes over an average commute period.

Data trustworthiness: Traditionally, if the sender of a message is trusted, then the content of the message is trusted as well. This notion is valid for long-lived, static trust relationships, but in VC systems there is often no ground for similar approaches. Moreover, to determine their trustworthiness, interaction with possibly adversarial data senders is hard due to the large-scale and fast-changing topology of VC networks. It is thus necessary to

assess the *trustworthiness of data per se*. The combination of multiple pieces of evidence or their absence allow nodes to decide on the fly on the trustworthiness of data, as obtained by other nodes in the VC system.

Secure localization: Location information is critical for VC systems, basically for all safety applications but also for position-based information dissemination. The determination of a node's own position is paramount, with global navigation satellite systems (GNSS) playing already a central role. However, the adversary can interfere, injecting forged or replayed navigation messages, and manipulate the location of the victim nodes. Defense mechanisms can allow receivers to detect adversarial GNSS transmissions and reject false location information. Alternatively, other dedicated infrastructure could be deployed to provide secure localization. To defend against adversarial nodes that falsely report their position, data consistency checks, or secure position verification techniques could be used.

Open Problems

Recent developments provide mature and deployable solutions for secure VC systems, although without addressing all issues. Three main future directions are briefly discussed here. For more information on secure and privacy-enhancing VC systems, their performance and effectiveness, and ongoing and future research, two recent articles are recommended as additional reading [1, 2].

Large-scale experimentation: Benchmarks, implementation of secure VC software, large-scale simulations, and field demonstrations show that secure VC is practical. With the appropriate design, secure VC systems can be as effective as their unsecured counterparts, which do impose the security overhead but are not an option for future deployment. Nonetheless, large-scale experimentation with any type (secure or not) of VC system has yet to be performed. Large-scale test beds for a thorough validation of the system will be beneficial toward secure VC system deployment.

Integration of VC systems with other systems: VC systems will closely interact and often be integrated with other systems. On the one hand, *commodity devices*, such as mobile phones, iPods, or (portable) navigation systems, would be used inside the vehicle and often be connected to the vehicle electronics. On the other hand, *cellular, WiFi, mesh, and wireless sensor networks* could be used for various services to VC nodes (e.g., Internet connectivity, local environmental information). Security solutions that consider explicitly this integration would be necessary.

Privacy: Existing solutions protect location privacy against adversaries with relatively limited physical presence. But dedicated adversaries, with knowledge of the geographical area and traffic and possibly other "offline" information, could identify user trajectories in their surveillance area. Other systems integrated with the VC systems may also "leak" private information. Worse even, infrastructure such as the closed circuit TV cameras installed already in many cities might render privacy-enhancing mechanisms for VC largely irrelevant. A careful investigation of all these aspects is needed to determine the level of privacy users can expect.

Recommended Reading

1. Papadimitratos P, Buttyan L, Holczer T, Schoch E, Freudiger J, Raya M, Ma Z, Kargl F, Kung A, Hubaux JP (2008) Secure vehicular communication systems: design and architecture. IEEE Communications Magazine 46(11):100–109, November 2008
2. Kargl F, Papadimitratos P, Buttyan L, Müter M, Wiedersheim B, Schoch E, Thong TV, Calandriello G, Held A, Kung A, Hubaux JP (2008) Secure vehicular communication systems: implementation, performance, and research challenges. IEEE Communications Magazine 46(11):110–118, November 2008
3. The Car-to-Car Communication Consortium, URL: http://www.car-to-car.org/index.php?id=1
4. IEEE 1609.2 (2006) IEEE trial-use standard for wireless access in vehicular environments – security services for applications and management messages, July 2006

Secure Wireless Mesh Networks

►Secure Network Coding for Wireless Mesh Networks

Secure Wireless Multicast

►Security of Group Communication in Wireless Mesh Networks

Security

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Information Theory

Definition

The security of ►encryption against unauthorized decryption, unauthorized changing of the data, etc. Security

should depend completely on the key. One distinguishes between the following two types of security:

- *Computational security*: Quantitative security is a quantitative security against unauthorized decryption, based on particular (usually mathematical) assumptions like the inherent difficulty of factoring sufficiently long numbers. Often a *security parameter* denotes the computational level of the security.
- *Unconditional security*: Security against unauthorized decryption assuming that the cryptanalyst has unlimited computing facilities (so, security in an ►information theoretic sense).

Recommended Reading

1. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton

Security Architecture

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition

The Security Architecture is an integral and critical component within the overall architecture of an enterprise, service, product, or application. It specifies the features and artifacts needed to protect confidentiality, integrity, and availability of information.

Theory

A Security Architecture can be produced to requirements (quality) as well as maintained over the period of its useful life (change). The design artifacts describe how the security measures/controls are positioned and how they relate to the overall IT architecture. They describe the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time. The security measures serve the purpose to maintain the system's quality attributes, among them confidentiality, integrity, availability, accountability, and assurance. They are specified to fulfill the security requirements that are derived from a risk analysis taking into account business, technical and legal/regulatory requirements.

Security requirements are often considered as non-functional requirements when systems are designed. In other words, they are not required for the system to meet its functional goals but are needed for a given level of assurance that the system will perform to meet the functional requirements that have been defined.

In recent years, there has been a trend toward a hierarchy of control objectives, controls and specific technical implementations of controls, which are implemented within a given security architecture in order to meet the security requirements.

Applications

Examples of security architectures based on public key techniques include ►X.509 and ►EMV. As part of the security architecture, a number of supporting services operated by so-called ►Trusted Third Parties may be defined, such as Registration Authorities, ►Certification Authorities, and ►Time Stamping Authorities. These are entities that are not part of the original system as such, but are introduced to offer security services in support to the security architecture.

Other examples are ►Kerberos, based on conventional cryptography and bespoke key management architectures, e.g., to handle online PIN-code (►Personal Identification Number) verification, which is characterized by key hierarchies, starting with session keys, or data keys at the bottom, which are protected or exchanged by key encryption keys, perhaps comprising several layers, and the top layer consisting of the so-called master keys.

Recommended Reading

1. http://www.opensecurityarchitecture.org/cms/definitions/it-security-architecture
2. http://www.owasp.org
3. http://www.sans.org
4. https://www.securityforum.org

Security Evaluation Criteria

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Common Criteria; ►FIPS 140-2 – Crypto Module Validation Program; ►ISO – International Organization for Standardization; ►SEC – System Evaluation Criteria; ►Security Evaluation Criteria

Definition

Security Evaluation Criteria are usually presented as a set of parameter thresholds that must be met for a system to be

evaluated and deemed acceptable. These criteria are established based on a Threat Assessment to establish the extent of the data sensitivity, the security policy, and the system characteristics.

Background

Security Evaluation Criteria are usually presented as a set of parameter thresholds that must be met for a system to be evaluated and deemed acceptable. These criteria are established based on a Threat Assessment to establish the extent of the data sensitivity, the security policy, and the system characteristics. The system is evaluated, the evaluation is measured against the criteria, and then an assessment is made of whether or not the system security characteristics meet the requirements as specified by the Security Evaluation Criteria. Criteria are typically unique to each system, the environment it is in, and how it is used. Important past frameworks of Security Evaluation Criteria have been the following:

- ***TCSEC by US Department of Defense* (1985)**: The Trusted Computer System Evaluation Criteria (TCSEC) is a collection of criteria that was previously used to grade or rate the security offered by a computer system product. No new evaluations are being conducted using the TCSEC although there are some still ongoing at this time. The TCSEC is sometimes referred to as the "Orange Book" because of its orange cover. The current version is dated 1985 (DOD 5200.28-STD, Library No. S225,711). The TCSEC, its interpretations, and guidelines all have different color covers and are sometimes known as the "Rainbow Series" [1]. It is available at http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html.
- ***ITSEC by the European Commission* (1991)**: The Information Technology Security Evaluation Criteria (ITSEC) is a European-developed criteria filling a role roughly equivalent to the TCSEC. Although the ITSEC and TCSEC have many similar requirements, there are some important distinctions. The ITSEC places increased emphasis on integrity and availability, and attempts to provide a uniform approach to the evaluation of both products and systems. The ITSEC also introduces a distinction between doing the right job (effectiveness) and doing the job right (correctness). In so doing, the ITSEC allows less restricted collections of requirements for a system at the expense of more complex and less comparable ratings and the need for effectiveness analysis of the features claimed for the evaluation.
- ***CTCPEC by CSE Canada* (1993)**: The Canadian Trusted Computer Product Evaluation Criteria is the Canadian equivalent of the TCSEC. It is somewhat more flexible than the TCSEC (along the lines of the ITSEC) while maintaining fairly close compatibility with individual TCSEC requirements.
- ***Common Criteria ISO 15408* (2001)**: In 1990, the Organization for Standardization (ISO) sought to develop a set of international standard evaluation criteria for general use. The CC project was started in 1993 in order to bring all these (and other) efforts together into a single international standard for IT security evaluation. The new criterion was to be responsive to the need for mutual recognition of standardized security evaluation results in a global IT market. The common criteria combine the best aspects of TCSEC and ITSEC and aims to supersede both of them.

Theory/Applications

Selection of all the relevant criteria using a threat/risk analysis as a guide is typically the first aspect. Typically, in order to mitigate the range of risks multiple criteria will need to be selected and evaluations performed. Some of the choices for security criteria are included below along with the category of the system in which they are applicable to:

- Components/Modules
  - Algorithm Implementations
    - Federal Information Processing Standards, i.e., FIPS 186-3
    - Cryptographic Algorithm Validation Program
  - Module Implementations
    - Federal Information Processing Standards, i.e., FIPS 140-2
    - Cryptographic Algorithm Validation Program
  - Side Channel Threats
    - Simple Power Analysis
    - Differential Power Analysis
    - Timing Attacks
    - Extreme Conditions, i.e., temperature, voltage, frequency, and radiation
- Commercial Off the shelf Items (COTS)
  - Personal Computers, Applications, and Appliances, i.e., routers and firewalls
    - Common Criteria ISO 15408
      - Security Features and Assurance
    - Security Content Automation Protocol
      - Secure Setup and Configuration of products
- Systems – Integration of COTs and custom component to full a system mission

- FISMA
  - FIPS 199, FIPS 200 and special publications
    - Process to assess and manage system risk
- ISO 17799
  - Risk Analysis and management of integrated systems
- Penetration Testing
  - Assessing the actual system as implemented resistance to penetration

Open Problems

The most difficult process is performing a objective and thorough risk assessment to choose the most relevant criteria to use as a evaluation standard.

Experimental Results

All processes are well established with known and predictable results and repeatability.

Recommended Reading

1. Trusted product evaluation program: computer security evaluation FAQ. http://www.radium.ncsc.mil/tpep/process/faq.html
2. www.commoncriteria.org,http://csrc.nist.gov/cc

Security for Mashups

Sachiko Yoshihama
IBM Research-Tokyo, Yamato, Kanagawa, Japan

Related Concepts

►Cross-Site Scripting (XSS) Attacks; ►Mashups

Definition

Security for mashups involves technologies or methods that allow integrating data or services from different sources, while protecting the security of each Web application or service from each other.

Background

Mashup is a technique for Web applications that integrates data or functions from different sources to create a new service. An example of a mashup application is a Web-based restaurant guide, which integrates an online map service to show the locations of the restaurants, combined with blog entries that provide reviews of those restaurants. The mashup approach allows creating new applications by leveraging content and services in existing Web sites.

However, a mashup application may pose new security risks, because typical mashup applications integrate into single Web page content and services from locations with different trust levels. Hence, if any of the sources have malicious intent, then that source might do something like injecting malicious client-side JavaScript code to attack the applications from other Web site (which is one form of cross-site scripting (XSS) attack).

Here are some examples of what a malicious script can do when injected into such integrated content. Such a malicious script may (1) steal sensitive information, such as passwords or credit card numbers as the user enters them into input fields provided by other applications; (2) modify information in parts of the Web page that appear to belong to other applications to display false or misleading information; (3) redirect the user to a malicious Web site that is hosting a phishing attack; or (4) transmit session cookies to the attacker, allowing certain authenticated sessions to be hijacked.

Most of today's Web browsers use the *same-origin policy* as a de facto standard security model, and therefore try to prohibit suspicious communication between windows and frames associated with different domains. However, the mashup of content and services from different domains often calls for data exchanges, and many mashup applications wind up trying to bypass or evade the same-origin policy by using various server-side or client-side techniques.

The server-side mashup method aggregates content from several third parties on the server of the mashup Web application so that the user's Web browser regards all of the content as coming from a single domain, and the communication between the sources domains is actually taking place in the server. In contrast, the client-side mashup method uses various loopholes in the same-origin policy while creating the mashup application in the user's Web browser. A typical client-side Mashup might use <script> tags to import client-side JavaScript code for third-party services. Since the same-origin policy is applied only at the granularity of the HTML document loaded within a browser window or frame, the imported JavaScript is regarded as belonging to the same domain as the HTML code that imported it, and this allows the effective aggregation of content from different domains. In addition, though the same-origin policy is applied to an asynchronous network access by using the XMLHttpRequest API, it is not applied to the network accesses originating with <img> or <script> tags in HTML, and thus the client-side JavaScript code can also communicate with third-party servers by dynamically modifying the src attributes of these elements.

Once the content from different domains is rendered in the same browser window or frame, it is not possible with the current browser architecture to enforce access controls.

In today's browser architecture, all of the JavaScript within a single window or frame share the same execution context, and cannot limit the access to variables or functions. In addition, existing variables or functions can easily be overwritten as long as a variable or function with the same name is declared. Any content within a browser window or frame can access the browser document object model (DOM) tree for that window or frame and read or write information that may belong to other applications.

Today's mashup applications assume there are trusted relationships between the mashup providers and the service providers to maintain the security of the application. However, even if a particular third-party service provider is not malicious, it may be vulnerable to XSS or other attacks, and other mashed-up applications can then be at risk through that vulnerable service. In particular, the screen 'real estate' used by advertisements is often subleased to other providers, making the trustworthiness of the actual content provider unclear.

Theory

Several researchers are addressing mashup security problems on existing browser architectures with the same-origin policy.

Subspace [1] proposes a secure mashup mechanism that isolates each application in an HTML iframe (inline frame) and then allowing only limited communication between the iframes. In general, applications downloaded from different domains cannot communicate, but most browsers relax the same-origin restriction by allowing overwriting of the built-in document.domain property with a super-domain of the original domain name. For example, two applications from foo1.bar.com and foo2.bar.com can communicate if they both overwrite their own document.domain to bar.com.

SMash [2] also isolates applications by using iframes to enable secure mashups on existing browsers. It consists of a JavaScript framework that provides three-layer communication channels that allow cross-domain communication between iframes. SMash uses a method called "fragment communication," which leverages the fact that the built-in windows.location property cannot be read but can be overwritten by iframes that belong to different domains. SMash also provides security tokens to protect message integrity and guard against impersonations. Since the upper layers of the communication channels are independent of the implementation of the bottom layer, the fragment communication can be replaced with other mechanisms once the browsers support native cross-domain communications. SMash became an industry standard and was published as open source software as OpenAjax Hub 2.0 [3].

Open Problems

An essential cause of the mashup security problems is that the same-origin policy is no longer suitable to today's Web applications. The security of the same-origin policy was based on an assumption that the content originating from the same domain could be trusted. However, that assumption no longer holds, due to the prevalence of mashups and user-generated content. Content that appears to be from the same domain may actually be originating elsewhere.

There is work underway to redesign the browser architecture to explicitly support secure cross-domain communications. For example, W3C is designing the HTML5 specification as the next generation of HTML, in which a new "postMessage" function is proposed to support explicit communications between frames containing content downloaded from different domains [4].

In addition, the XMLHttpRequest Level 2 specification [5] proposes to extend XMLHttpRequest to support explicit cross-domain network communication. The access control specification [6] that [5] depends on is already implemented in Firefox 3.5 [7]. It has also been announced that Microsoft Internet Explorer® will support content isolation and explicit cross-domain communication [8].

However, mashup security issues originate from the fact that the architecture of the Web has changed much since the early days when the Internet was a relatively closed environment with few security risks. There are some ongoing efforts to radically change those legacy technologies to make them more secure, but in many cases such changes cannot be adopted because of backward compatibility requirements. It is an ongoing challenge to improve Web security while maintaining backward compatibility.

Recommended Reading

1. Jackson C, Wang H (2007) Subspace: secure cross-domain communication for Web mashups. 16th International World Wide Web Conference (WWW '07). Banff, Alberta, Canada, May 8–12, 2007
2. Keukelaere FD, Bhola S, Steiner M, Chari S, Yoshihama S (2008) SMash: secure component model for cross-domain mashups on unmodified browsers. 17th International World Wide Web Conference (WWW '08). Beijing, China, April 25–28, 2008
3. OpenAjax Hub 2.0, http://www.openajax.org/member/wiki/OpenAjax_Hub_2.0_Specification
4. HTML5, http://www.w3.org/html/html5/
5. XMLHttpRequest Level 2, http://www.w3.org/TR/XMLHttpRequest2/
6. Cross-Origin Resource Sharing, W3C Working Draft 17 March 2009, http://www.w3.org/TR/access-control/
7. HTTP access control, https://developer.mozilla.org/En/HTTP_Access_Control
8. Microsoft, Better Ajax Development: Windows® Internet Explorer® 8. Whitepaper, March 2008

Security Implication in Virtualization

Tae Oh[1], Shinyoung Lim[2], Young B. Choi[3], Jungwoo Ryoo[4]
[1]School of Informatics, Rochester Institute of Technology, Rochester, NY, USA
[2]School of Health and Rehabilitation Services, University of Pittsburgh, Pittsburgh, PA, USA
[3]Department of Natural Science, Mathematics and Technology, School of Undergraduate Studies, Regent University, Virginia Beach, VA, USA
[4]Division of Business and Engineering, Penn State Altoona, Altoona, PA, USA

Synonyms

Virtualization security

Related Concepts

►Operating Systems; ►Virtualization

Definition

Virtualization has many benefits such as server consolidation, lower energy bills, faster hardware, networking, and computing efficiencies. However, virtualization is still prone to security compromises by an attacker and has several significant security implications.

Background

What Is Virtualization?

Virtualization is the process of mapping the resources and interfaces of a virtual resource into the resources and interfaces of a host machine. This is a common strategy for improving the utilization of the existing computing resources [1]. Virtualization provides a logical view instead of the physical view of the computing resources and makes available multiple operating systems (OS) running in parallel in a single physical computing platform.

The virtualization efforts began in 1990s and were primarily for testing and evaluating the new software. Virtualization recreates the end-user environments in a single mainframe computer [1]. This way testers and evaluators can see how the software performs on Windows or Linux machines with various user settings.

When ×86 architecture and inexpensive PCs became available, the virtualization concept faded away [2]. However, the virtualization for ×86 platforms was developed and revived by a company named VMware [3]. VMware developed the first hypervisor for the ×86 architecture in 1990 and started the growth of the virtualization software industry.

Virtualization Types

Virtualization technology has been rapidly emerging as an important component of future computer systems, and currently, there are three types of virtualization, which are storage, network, and server virtualizations.

- *Storage virtualization* combines multiple storage devices into one storage resource so that storage appears to be a single device.
- *Network virtualization* divides available bandwidth into multiple channels, and each channel is assigned to a particular device and server to operate in real-time.
- *Server virtualization* provides a virtual version of a physical server and provides all server resources including processors, operating system, and memory virtually for the software running on the virtual server.

Out of these three different types of virtualization, server virtualization is the most popular and common technology. When people mention the term "virtualization," they are usually referring to server virtualization.

Virtualization Benefits

The utilization and implementation of virtualization has been increasing dramatically for industries, academia, and government agencies since there are many benefits of implementing virtualization. The list of benefits is provided below.

- Server Consolidation: This offers huge money savings to the companies and organizations by reducing the number and types of servers that supports their employees and applications. Currently, 60–80% of IT departments have been initiating consolidation projects since virtualization offers significant cost savings.
- Dynamic Load Balancing: This offers the distribution of workload to multiple devices and machines. Sometimes, applications running on a dedicated machine could slow down when the machine experiences increased workload. Dynamic load balancing helps to distribute the workload in such a way that applications do not experience a bottleneck or disruptions. This is very important for critical applications for businesses and organizations.
- Lower Power Consumption: The consolidation of machines and devices offers low power consumption.

Because of the lower number of machines to manage, power consumption for cooling the machines also decreases.
- Information Security: Beside the energy efficiency, virtualization offers information security by isolating processes from attackers and malware. The user access to applications can be controlled tightly, and the application can be isolated from the end-user applications.
- Failover Functionality: Virtualization has the benefit of maintaining a system without taking down the system. This can be achieved by containing the applications in a pool. A user should be able to switch between multiple virtual machines (VMs).

Virtualization Security Risks

Virtualization offers a confined area so that any function performed within a virtual machine (VM) cannot affect the host OS or physical machine. However, there are a few security risks to be considered when the host OS is infected or compromised. This can affect all VMs in the machine and intruders can modify VMs to have a negative effect.

Another security concern to consider is "hyperjacking," in which an attacker creates and develops a thin hypervisor, and can take control of the underlying operating system. A good example of hyperjacking is software called Blue Pill rootkit [4]. This software was developed by a security researcher named Joanna Rutkowska and evades all detection from system administrators including the people responsible for detecting any infection or abnormality. This evasiveness allows the toolkit to take control of the operating system easily.

Lastly, it is possible to capture packets from the layer 2(data link) of the network by configuring a network interface card (NIC) even in the virtualized environment. This allows a computer to control the data flow of other network-connected systems and could cause unauthorized disclosure or the theft of stored data and programs.

Theory

Operating System-Based Virtualization Methods

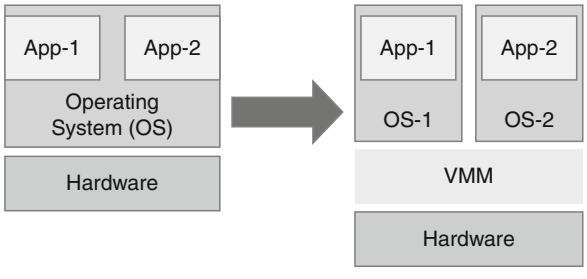
Virtualization for operating systems offers several benefits [2]. For example, upgrading OSes can be staged across VMs to minimize the downtime, and the failures in guest software can be isolated by the virtual machine monitor (VMM). Additionally, VMM has been improving recently, and the product offers wider benefits for clients and server systems. The following sections describe the three categories of OS-based virtualizations usages.

Workload Isolation

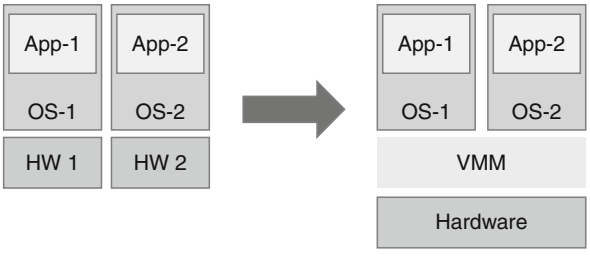
Workload isolation offers a way to stack the software on their own VMs. There could be multiple VMs, and each VM has a stack of software running on the assigned VM as shown in Fig. 1. This offers security and reliability advantages since the intrusion can be confined to the affected VM only. This category was developed by Thomas Bressoud and Fred Schneider by examining the fault-tolerance behaviors in separate VMs to recover from a system failure. Currently, workload isolation is offered in Microsoft's Next Generation Secure Computing Base (NGSCB) and VMware's Assured Computing Environment (ACE).

Workload Consolidation

Workload consolidation offers to consolidate the workloads into a single physical platform instead of distributing the workload to a number of servers as shown in Fig. 2. Currently, data centers are challenged by a large number of heterogeneous servers running the same OS and applications for file serving and web hosting. This creates inefficiency especially if many servers are being underutilized. Therefore, workload consolidation can provide easier management of the consolidated platform.



Security Implication in Virtualization. Fig. 1 Workload isolation [2]



Security Implication in Virtualization. Fig. 2 Workload consolidation [2]

Workload Migration

Workload migration offers to decouple the OS and applications from current hardware and migrate to different hardware as shown in Fig. 3. This method is used for hardware maintenance and workload balancing, and improves the quality of service (QoS) with lower operating cost. The workload migration has been demonstrated by the Xen and Internet Suspend-Resume Projects [4].

Hardware-Based Virtualization

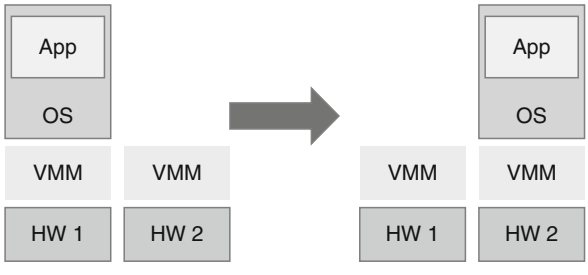
To solve the big performance overhead of virtualization, several processor manufacturing companies have been developing hardware-based virtualization technologies. Virtualizing a typical computer system involves the system's shared resources such as Central Processing Unit (CPU), Input/Output (I/O), and some specialized peripheral devices. Hypervisors accommodate virtualization using the hardware in addition to the software. Hardware-based virtualization technologies have the advantages and disadvantages in security.

There were several hardware-based virtualization products such as Intel Vanderpool and AMD Pacifica Technology, Intel LaGrande and AMD Presario Technology, and Trusted Platform Module (TPM) Virtualization [4].

Performance Comparisons

Operating systems (OS) virtualization has many advantages including transparent migration of applications, server consolidation, online OS maintenance, and enhanced system security. It also has its own share of challenges including system call interposition, virtualization state management, and race conditions.

The effectiveness of OS virtualization implementation was measured on both micro-benchmarks and real application workloads. It quantified race condition handling cost as a basis of comparison among different virtualization technologies. The hardware virtualization-based measurement details of the experimental results using a very low overhead Linux kernel module working across multiple kernel versions can be found in [5].



Security Implication in Virtualization. Fig. 3 Workload migrations [2]

System Architectures for Virtualization

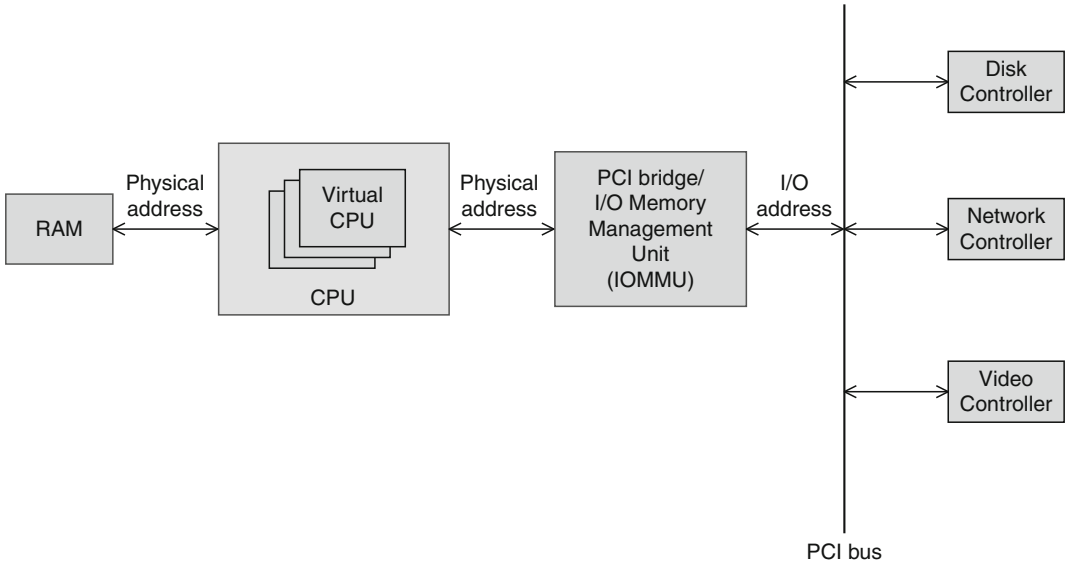
Virtualization from the computer architecture domain perspective is to present an environment to one layer in an information technology stack, which abstracts or represents a lower layer. System architects typically add this layer of indirection between existing layers in a system stack to address specific problems such issues as support for functionality, standardizing interfaces on logical models, or usage for transparent load balancing of shared resources. But the major disadvantage of virtualization is its excessive performance overhead. A single emulated machine instruction can easily expand to a large number of real instructions and be a major cause for significant performance degradation. To alleviate this problem, CPU manufacturers have been developing hardware support for virtualization, in which all of the emulation occurs in the CPU itself. Major vendors have also been working on virtualization capabilities for CPU, I/O, and other devices. Traditional computer systems consist of memory (Random Access Memory), a CPU, an I/O controller, and I/O devices such as a disk, network, and video controllers. Virtualizing the shared resources in this architecture involves changing the CPU and the bus controller that arbitrates accesses to the I/O bus. A typical CPU consists of many shared resources such as interrupt vectors, page tables, interrupt controllers, timers, and special descriptor tables. The virtualized system is able to virtualize these resources with the right processor abstractions. It must give the illusion of a VM to the guest operating system that is running inside the physical computing resources. This involves managing physical memory, interrupts, faults, and I/O devices [4, 5] (Fig. 4).

Virtualization Security Overview

Migration of computing resources to a virtualized environment has little or no effect on most of the resources' vulnerabilities and threats, and many virtualization features offer both advantages and disadvantages in security [1]. These security implications can be explained in three ways: (1) the isolation of guest OSes from each other, and the underlying hypervisor and the host OS, (2) guest OS monitoring, and (3) image and snapshot management.

Guest OS Isolation

The hypervisor manages guest OS access to hardware such as memory, CPU, and storage. Partitioning these resources for each guest OS implies that a VM can access its own resources but cannot access the other guest OSes' resources



Security Implication in Virtualization. Fig. 4 Virtualized system architecture overview [4]

or any resources not allocated for virtualization use. The resources used by guest OSes can be physically partitioned or logically partitioned. In physical partitioning, the hypervisor assigns separate physical resources such as disk partitions, disk drives, and NICs to each guest. In logical partitioning, the resources are assigned to a single host or across multiple hosts allowing multiple guest OSes to share the same physical resources such as RAM and processors [1].

Guest OS Monitoring

The hypervisor has introspection ability to monitor each guest OS as it is running. Introspection provides full monitoring capabilities for the elements including network traffic, processes, memory, and many other elements of a guest OS. The hypervisor provides information such as firewalling, intrusion detection, and access control through introspection. In guest OS monitoring, network traffic monitoring is especially important when networking is performed, and network traffic monitoring can be done between two guest OSes on the host or between a guest OS and the host OS [1].

Image and Snapshot Management

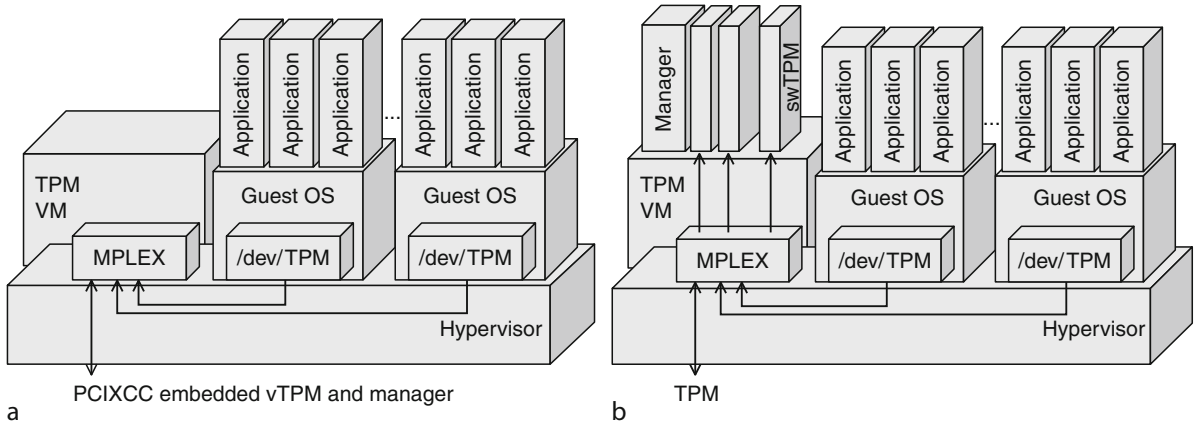
A simple act of creating guest machine images and snapshots does not affect the vulnerabilities in the guest OSes, and their services and applications. However, images and snapshots themselves can affect security both positively and negatively. It can also affect information technology operations. The most important security issue with images and snapshots is that they contain very sensitive data such as personal data and passwords. The proliferation of images referred to as sprawl is also another important problem resulting from increasing the use of virtualization [1]. It is necessary for the organizations to maintain a good set of images and snapshots for testing and eventual efficient management of virtualization.

Access Control Policy-Based Virtualization Security

As one of the representative hypervisor security architecture, the major components of the sHype Access Control Architecture is composed of the PM (Policy Manager), ACM (Access Control Module), and MH (Mediation Hooks). PM delivers the services of securely creating and updating the access control policy to decide which VMs can access which resources. ACM performs resource control, access control between VMs, isolation of virtual resources, and Trusted Platform Module (TPM)-based attestation. Security MHs mediate access to resources inside the hypervisor, enabling information flow between VMs [4]. Figure 6 shows the sHype hypervisor security architecture [4].

Trusted Platform Module-Based Virtualization Security

Trusted Platform Module (TPM) provides security that software cannot compromise [4]. TPM is an architecture built on top of a cryptographic coprocessor and



MPLEX: Mutiplexer TPM: Trusted platform module
vTPM: Virtual TPM swTPM: Software TPM VM: Virtual machine
PCIXCC: IBM PCIX-Crypto coprocessor

Security Implication in Virtualization. Fig. 5 TPM Virtualization built on top of (**a**) IBM PCIX-Cryto Coprecessor and (**b**) a Hardware TPM [4]

hardware TPM as shown in Fig. 5. This architecture allows one to use a TPM on systems running multiple operating systems simultaneously. PCIX Crypto Co-processor (PCIXCC) has tamper-sensing and responding mechanisms, and this is a great platform for virtual PM functionality. PCIXCC prevents intruders from accessing sensitive data on the device. Virtual TPM, as shown in Fig. 5b, protects the system by controlling multiple isolated VMs. This will enable multiple VMs to be housed in a single platform.

Applications

Overview

The motivation of information security for virtualization originates from the concept of separation of all resources from the virtual computer. With this separation, users can control interactions with these virtual computers using a small security kernel, and they can run whatever OS and software. As one computer gets too heavily loaded, they can also balance the workloads by moving the work to other virtual machines, which will prevent denial of service and other overloading issues. All of the issues one has with operating systems, security kernels, perimeter controls, intrusion detection and response, and encryption will still be there. In addition, users will not have the same control over where things happen, and users will lose the ability to keep track of things even more. There are a few commercially available security solutions for virtualization-based service systems [6].

Hypervisor Architecture

Early VMs were essentially computing environments that emulated the host system's existing hardware while arbitrating access to shared system resources. Systems of the time (i.e., largely mainframes designed in the 1960s and 1970s, such as IBM VM/370's predecessors) were expensive. These early VMs gave companies distinct economic advantages by allowing them to partition one physical system into multiple logical systems. The same advantages exist today for the IBM VM/370's heir, such as IBM PR/SM and z/VM VM monitors (VMMs), and the S/390 and zSeries mainframes. Hypervisors are particularly well suited for a few basic but strong security primitives such as separation and controlled sharing. Hypervisors function as reference monitors, providing workload isolation on an operating system instance granularity (as opposed to operating systems that strive to provide process-level isolation). The reference monitor mediates all security-sensitive operations such as access to objects or communications between VMs. A hypervisor is also a key element in the trusted computing-based hardware, firmware, and software components in layered-system architectures that must be corrected to enforce the explicit or implicit system security policy. Because the hypervisor reference monitor mediates access to the entities such as processors, memory, disks, and VMs, hypervisors are often orders of magnitude smaller and less complex than modern operating systems. Hypervisors have traditionally supported strong isolation or separation of VMs and their workloads, including fault isolation – limiting

an application or operating system fault's effect within a VM. However, hypervisors that support sharing based on explicit security policies and labels associated with each VM and its resources were not available on a large scale. This is partly because of the commercial requirements and the systems' high assurance government/military-oriented design targets. Because hypervisors must maintain isolation while maximizing resource utilization and support for the more distributed and interconnected nature of contemporary workloads, policy-driven controlled sharing requirements for commercial hypervisors on mid-range and high-volume systems are increasingly challenging.

Secure Hypervisor (sHype) Example

Figure 6 shows the sHype hypervisor security architecture developed by IBM and its integration into a VMM environment with major components of policy manager, which oversees the access-control policy, the access-control module (ACM), and the meditation hooks. The major goal of sHype is the establishment of a secure foundation for server platforms, functions of strong isolation and meditated sharing between VMs, attestation and integrity assurances, resource control and accurate accounting management, and secure services.

The security of a full virtualization solution is heavily dependent on the individual security of each of its components, including the hypervisor, host computer, host operating system, guest operating systems, applications, and storage. Organizations should secure all of these elements and maintain their security based on the security policies. Virtualization can be used in many ways. Therefore, the appropriate security controls for each situation vary. In general, organizations should have the same security controls in place for the virtualized operating systems as they have for the operating systems running directly on hardware. The same is true for applications running on guest operating systems. If the organization has a security policy for an application, it should apply the same regardless of whether the application is running on an operating system within a hypervisor or on an operating system running on physical hardware [1, 2, 4, 6] (Table 1).

Commercially Available Products

The following table describes commercially available virtualization products and their security solutions.

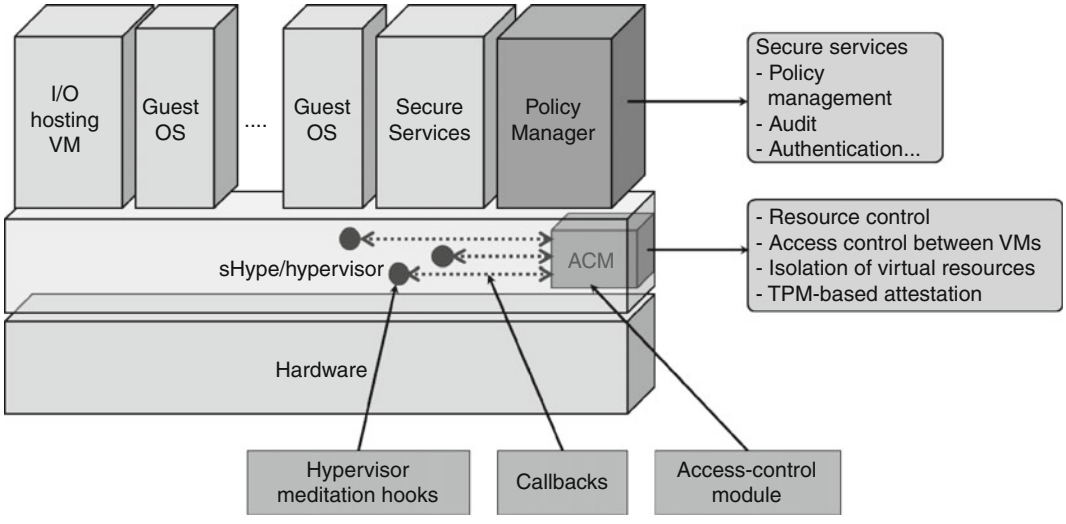
Virtualization Security Tools

There are not many virtualization security tools. Vendors are rarely able to release additional virtualization security tools. Currently, the tools work with only one platform since the virtualization and APIs are relatively new. Table 2 summarizes a few representative virtualization security tools.

The other companies that offer virtualization security products are Altor Networks, Catbird Networks, Fortisphere, and Montego Networks.

Open Problems and Future Directions

Full virtualization has many benefits including efficiency and energy savings but has a few negative implications.



Security Implication in Virtualization. Fig. 6 The sHype security architecture [4]

Security Implication in Virtualization. Table 1 Commercially available virtualization products and their security solutions

Type of virtualization	Commercial product/solution	Remarks
Hypervisor type	IBM VM/370s (IBM PR/SM, z/VM VM monitors, S/390, zSeries mainframes, and DEC VAX VMM, KVM/370 [7–10]	Early primitive versions of virtualization by VMs for computing environments that stimulated or emulated the host system's existing hardware while arbitrating access to shared system resources
Hardware virtualization support type	Intel Vanderpool, AMD Pacifica Technology (Secure VM), and IBM Power5 [11–13]	Encapsulation all of the components of a guest operating system, including its applications and the virtual resources they use, into a single logical entity
I/O Memory management type	Intel LaGrande and AMD Presidio Technology [14–19]	Process isolation, sealed storage, platform attestation, and secure I/O paths as the common core design concepts
sHype type	Xen open source hypervisor, IBM Research's Trusted Virtual Data Center, and Access control approach (Policy Manager) [20, 21]	Establishment of a secure foundation for server platforms, providing functions of strong isolation and mediated sharing between VMs; attestation and integrity guarantee; resource and account controls; and secure services
Trusted platform module (TPM) type	PCIXCC (IBM PCIX-Crypto Coprocessor) and vTPM (virtual TPM) [22]	Emerging security building block offering the system-wide hardware roots of trust that the system software can not compromise
Partition of tamper-resistant hardware type	Terra [23]	Creation of protected systems by partitioning a tamper-resistant hardware into multiple VMs, presenting multiple virtual hosts on a single, general purpose platform

Security Implication in Virtualization. Table 2 Some representative virtualization security tools

Virtualization Security Tool	Description
Apani's EpiForce VM	Uses VPN and network security such as intrusion prevention systems and firewalls
Blue Lane Technologies' Virtual Shield	Supports only VMWare™ system Protects against known vulnerabilities by identifying the possible issues
Reflex Security's vTrust	Supports VMWare™ ESX and Citrix's Xen virtualization Provides intrusion detection, firewalls, antispyware, and network policy enforcement Administrators are able to view the operation of virtual machines

Virtualization adds another layer of technology, and this layer adds complexity to security management. Additional security controls are needed to handle the virtualization features. Also, consolidating a number of systems or servers into one physical platform can create a large problem if the security is compromised on this platform. If the intruder breaks the security of the host OS, the intruder will have access to all systems and servers. Another problem is that some virtualization is designed to share information between the systems. Sharing information could be dangerous when the information is shared between the intruders. Lastly, some virtualized environments are quite dynamic, which could be difficult for security management and can blur the security boundary between VMs.

Virtualization Security Problems

There are several security problems in virtualization-based systems and networks. Especially, network-based security systems usually do not track communications between VMs on the same physical machine. This can cause severe security problems. In general, many organizations overlooked security in a rush to implement virtualization [24].

Some of the major virtualization security problems include hypervisor vulnerability, OS vulnerabilities, and configuration-management problems. The most severe security problem in virtualization is that hypervisors have more access to hardware resources than typical applications do. As a result, the attackers can obtain access to the hypervisor and eventually multiple VMs running on the hypervisor. Traditional OSes with vulnerabilities can be exploited easily by attackers. Managing many VMs on multiple host computers in an organization according to an organizational security policy is extremely difficult. If keeping track of these and taking necessary proper actions are not successful, this will make it possible for the attackers to take advantage of the system vulnerabilities [24].

Future Directions for Virtualization-Based Security

The goal of realizing simplified operational security management for information services will drive progress and innovation across the spectrum, from low-level hardware-related developments to high-level distributed systems management. Heterogeneous multi-core processors are emerging in the high-volume market and chip designers will integrate security and trusted platform-based computing functions (such as cryptographic acceleration engines and TPMs) into the processor design. As hypervisors mature, processor support for recursive virtualization might be necessary to preserve the investment made in these mature solutions. An increasing number of peripherals will likely support self-virtualization. For instance, they will be able to support multiple logical adaptor instances within one physical adaptor [4, 24].

Peripherals will also support trusted computing authentication and integrity goals with the incorporation of TPMs and attestation capabilities. The combination of these developments will lead to the emergence of peripherals that can enforce system-wide access control and information flow security policies, thus becoming an extended part of the trusted computing base while relieving the hypervisor of the policy-enforcement obligations associated with these resources [25].

Utility and other distributed computing models, such as cloud computing, will also continue to gain acceptance. This increase is partly due to the economic advantages of having access to essentially unlimited computing resources, paying only for what one uses, and being able to stipulate service-level agreements or quality-of-service guarantees with a small upfront investment. Such infrastructure usage will require reliable and secure resource monitoring that both workload and infrastructure owners can trust. The hardware support for resource control will also be needed to enforce resource usage limits that will defend the system against denial-of-service attacks in mixed-use environments. The possibility of implementing a distributed trusted computing base, built upon the secure hardware and virtualization foundations, is the great promise of these technologies. The academic researchers and industry communities must leverage emerging trusted computing technologies and virtualization capabilities to further bridge the middleware-to-systems gap and relieve application developers of the burden of implementing and verifying security-related functionality [26, 27].

Conclusion

Virtualization offers many benefits. Interest in virtualization technologies is growing rapidly. This trend is mainly due to the fact that many organizations are deciding to increase the efficiency of their computing resources by consolidating them through virtualization to improve efficiency. The consolidation efforts also reduce the cost, which is another plus. However, they introduced some new security problems.

This entry explained the security implications in virtualization and their countermeasures with some examples of emerging solutions. In addition, the future directions of the virtualization security research are discussed. The use and adoption of virtualization technologies will keep growing in the foreseeable future to provide computing efficiency necessary for most of organizations to survive. Therefore, it is an absolute imperative to develop the security technologies that go along with the advancement of the virtualization technologies.

Acronyms and Abbreviations

ACE	Assured Computing Environment
ACM	Access Control Module
API	Application Programming Interface
CPU	Central Processing Unit
DoS	Denial of Service
FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
IDPS	Intrusion Detection and Prevention System
I/O	Input/Output
IP	Internet Protocol

IT	Information Technology
ITL	Information Technology Laboratory
JVM	Java Virtual Machine
LAN	Local Area Network
MAC	Media Access Control
MH	Mediation Hooks
NAPT	Network Address and Port Translation
NAS	Network Attached Storage
NAT	Network Address Translation
NGSCB	Next Generation Secure Computing Base
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
OMB	Office of Management and Budget
OS	Operating Systems
OVF	Open Virtualization Format
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCIXCC	IBM PCIX-Crypto Coprocessor
PII	Personally Identifiable Information
PM	Policy Manager
QoS	Quality of Service
RAM	Random Access Memory
SAN	Storage Area Network
SLA	Service Level Agreement
SP	Special Publication
TPM	Trusted Platform Module
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Monitor
VPN	Virtual Private Network
vTPM	Virtual Trusted Platform Module

Recommended Reading

1. Scarfone K, Souppaya M, Hoffman P (2010) Guide to security for full virtualization technologies (draft). National Institute of Standards and Technology, Special Publication 800-125 (draft), July 2010
2. Uhlig R et al (2005) Intel virtualization technology. Computer 38(5):48–56
3. VMWare (2010) http://www.vmware.com/
4. Perez R, van Doorn L (2008) Virtualization and hardware-based security. IEEE Security Privacy 6(5):24–31
5. Laadan O, Nieh J (2010) Operating system virtualization: practice and experience. In: SYSTOR 2010, Haifa, Israel, 24–26 May 2010
6. Cohen F (2010) The virtualization solution. IEEE Security Privacy 8(3):60–63
7. Karger PA et al (1991) A retrospective on the VAX VMM security Kernel. IEEE Trans Softw Eng 17(11):1147–1165
8. Gold BD, Linde RR, Cudney PF (1984) KVM/370 in retrospect. In: Proceedings of IEEE symposium security and privacy. IEEE CS Press, New York, pp 13–23
9. Robin JS, Irvine CE (2000) Analysis of the intel pentium's ability to support a secure virtual machine monitor. In: Proceedings of 9th usenix security symposium. Usenix Assoc, Berkeley, p 10
10. Intel, Intel 64 and IA-32 architectures software developer's manual, volume 3B: system programming guide. www.intel.com/products/processor/manuals/
11. AMD, AMD64 architecture programmer's manual, volume 2: system programming, http://developer.amd.com/documentation/guides/Pages/default.aspx.
12. Armstrong WJ et al (2005) Advanced virtualization capabilities of power5 systems. IBM J Res Dev 49(4/5):523–532
13. Advanced micro devices, AMD I/O virtualization technology (IOMMU) specification, 2006. www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf
14. Grawrock D (2006) The intel safer computing initiative: building blocks for trusted computing. Intel Press, Hillsboro
15. LaPedus M (2004) AMD tips 'Pacifica' and 'Presidio' processors for'06. www.eetimes.com/semi/news/showArticle.jhtml?articleID=52601317. Accessed Nov 2004
16. Trusted computing group (2004) TCG specification architecture overview, revision 1.2, www.trustedcomputinggroup.org/downloads/TCG_1_0_Architecture_Overview.pdf. Accessed April 2004
17. Sailer R et al (2005) Building a MAC-based security architecture for the Xen open-source hypervisor. In: Proceedings of the 21st annual computer security applications conference (ACSAC), IEEE CS Press, Washington, DC, pp 276–285
18. Barham P et al (2003) Xen and the art of virtualization. In: Proceedings of 19th ACM symposium operating systems principles. ACM Press, New York, pp 164–177
19. Sailer R et al (2004) Design and implementation of a TCG-based integrity measurement architecture. In: Proceedings of 13th usenix security symposium. Usenix Association, Hillsbro, pp 223–238
20. Berger S et al (2008) TVDc: managing security in the trusted virtual datacenter. ACM SIGOPS Oper Syst Rev 42(1):40–47
21. Dyer J et al (2001) Building the IBM 4758 secure cryptographic coprocessor. Computer 34(10):57–66
22. Berger S et al (2006) vTPM – virtualizing the trusted platform module. Proceedings of 15th Usenix security symposium. Usenix Association, Hillsbro, pp 305–320
23. Garfinkel T et al (2003) Terra: a virtual machine-based platform for trusted computing. Proceedings of ACM symposium operating system principles. ACM Press, New York, pp 193–206
24. Vaughan-Nichols SJ (2008) Virtualization sparks security concerns. Computer 41(8):13–15
25. Ray E, Schultz E (2004) Virtualization security. 2004
26. IBM processor resource/systems management (PR/SM) planning guide, SB10-7036-01, eServer zSeries 990
27. Anderson JP et al (1972) Computer security technology planning study, tech. report ESD-TR-73-51, vols I and II, Air Force Systems Command, USAF. Electronic Systems Division, Bedford
28. Sharif M, Lee W, Cui W, Lanzi A (2009) Secure in-VM monitoring using hardware virtualization, CCS'09, Chicago, IL, 9–13 Nov 2009
29. Creasy RJ (1981) The origin of the VM/370 time-sharing system, IBM J Res Dev 25(5):483–490

Security of Cognitive Radios

Jung-Min "Jerry" Park, Kaigui Bian
Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA

Related Concepts

►Dynamic Spectrum Access

Definition

Security issues pertinent to the unique characteristics of cognitive radios – namely, their ability to dynamically change their transmission or reception parameters based on the observations in the external and internal radio environments.

Background

The concept of cognitive radios was first presented officially by Joseph Mitola III and Gerald Q. Maguire, Jr. in an article published in 1999 [1]. At the time, cognitive radio was a novel paradigm for wireless communications that Mitola later described as:

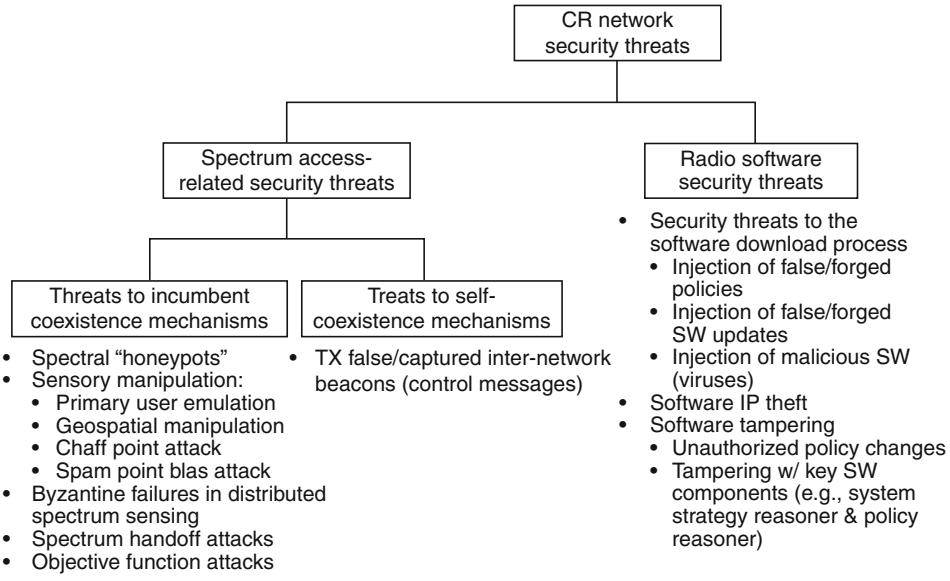
> The point in which wireless personal digital assistants (PDAs) and the related networks are sufficiently computationally intelligent about radio resources and related computer-to-computer communications to detect user communications needs as a function of use context, and to provide radio resources and wireless services most appropriate to those needs.

The emerging area of cognitive radio security has attracted significant attention from both academia and industry, and several works have been published recently [2–6].

Applications

The successful deployment of cognitive radio networks and the realization of their benefits depend on the placement of critical security mechanisms in sufficiently robust form. The emergence of cognitive radio technology and the dynamic spectrum sharing paradigm raises new security implications that have not been studied previously. Researchers have only recently begun to examine the security issues specific to cognitive radio devices and networks.

Spectrum coexistence, or simply coexistence, is an important attribute of cognitive radio networks. Cognitive radio networks support two types of coexistence: incumbent coexistence (i.e., coexistence between primary and secondary networks) and self-coexistence (i.e., coexistence between secondary networks). Cognitive radio networks must employ coexistence mechanisms of both types in order to perform dynamic spectrum access without causing harmful interference. Adversaries can exploit vulnerabilities in the coexistence protocols or mechanisms to



Security of Cognitive Radios. Fig. 1 Taxonomy of security threats to cognitive radio systems and networks

attack cognitive radio networks. Figure 1 provides a taxonomy of security threats to cognitive radio networks. Note that this taxonomy focuses on "active" threats unique to cognitive radio networks. Passive threats, such as eavesdropping, and threats also applicable to conventional wireless (i.e., noncognitive) networks are not included in the taxonomy. The figure categorizes the threats into two broad categories: spectrum access-related security threats and radio software security threats. The former can be further classified into two subcategories: threats to incumbent coexistence and threats to self-coexistence.

There are several attacks that exploit incumbent coexistence, including primary user emulation, geospatial manipulation, and spectrum handoff attacks. Among these attacks, primary user emulation has attracted particular attention from the research community, and a number of countermeasures have been proposed. In a primary user emulation attack, a rogue secondary user attempts to gain priority over other secondary users by transmitting signals that emulate the characteristics of the primary users' signals [2].

Self-coexistence mechanisms for a cognitive radio network are defined as part of the network's air interface and have features specific to that air interface. Ensuring the congruous coexistence of cognitive radio networks (i.e., harmonious self-coexistence) is of critical importance to minimize self-interference between neighboring networks and perform dynamic resource sharing. For instance, IEEE 802.22 addresses self-coexistence using inter-base station dynamic resource sharing and prescribes two supporting mechanisms: nonexclusive spectrum sharing and exclusive spectrum sharing. Unfortunately, the various self-coexistence mechanisms employed by cognitive radio networks are vulnerable to attacks. In [6], Bian and Park discuss some of these attacks.

The advantages of cognitive radios can be offset by the lack of security and reliability of the underlying software, which serves as the control and command for the radio system. In particular, the programmability of the cognitive radio devices raises serious security concerns. Without proper software protection mechanisms in place, cognitive radios are vulnerable to a host of attacks targeting the radio software. These attacks may include execution of malicious code, removal of software-based authentication or access control functions, intellectual property theft via reverse engineering, disruption of radio software reconfiguration, etc.

Recommended Reading

1. Mitola J III, Maguire GQ Jr (1999) Cognitive radio: making software radios more personal. IEEE Pers Commun 6(4):13–18
2. Chen R, Park J, Reed J (2008) Defense against primary user emulation attacks in cognitive radio networks. IEEE J Sel Areas Commun 26(1):25–37
3. Chen R, Park J, Hou T, Reed J (2008) Toward secure distributed spectrum sensing in cognitive radio networks. IEEE Commun Mag 46(4):50–55
4. Clancy T, Goergen N (2008) Security in cognitive radio networks: threats and mitigation. In: Proceedings of the international conference on cognitive radio oriented wireless networks and communications, Singapore
5. Brown TB, Sethi A (2008) Potential cognitive radio denial-of-service vulnerabilities and protection countermeasures: a multi-dimensional analysis and assessment. J Mobile Netw Appl 13(5):516–532
6. Bian K, Park J (2008) Security vulnerabilities in IEEE 802.22 (Invited Paper). In: Proceedings of the fourth international wireless internet conference (WICON), Maui

Security of Distance Bounding Protocols

Srdjan Capkun
System Security Group, Department of Computer Science, ETH Zurich, Zürich, Switzerland

Definition

Distance bounding refers to a class of protocols where a verifying party (verifier) measures an upper bound on the physical distance to the proving party (prover).

Background

Distance bounding protocols were first introduced by Stefan Brands and David Chaum as a solution to distance fraud and mafia fraud attacks [1]. Distance fraud attacks are attacks in which a prover pretends to be closer to the verifier than it really is, whereas mafia fraud attacks are those in which an attacker impersonates a party by relaying messages from that party to the verifier. A number of distance bounding protocols later followed that specifically considered additional security properties like location privacy [4], and performance issues like robustness to message loss [3] and efficient mutual distance bounding [2]. The feasibility of the implementations of distance bounding protocols was investigated in the context of ultrasonic [6] and radio communication [5].

Theory

Distance bounding protocols are challenge–response protocols with round-trip time measurements. The main component of distance bounding protocols is a time-critical phase in which the verifier sends a challenge to the prover,

based on which the prover computes a reply which it then immediately sends back to the verifier. The verifier measures the time between sending the challenge and receiving the reply; based on this time it estimates the distance to the prover. The challenge is chosen randomly and is unpredictable to the prover and to any third party. Distance fraud and Mafia fraud attacks are prevented since (1) the challenge is chosen randomly and is unpredictable to the prover and to any third party and (2) the prover cannot construct the correct reply before it receives the challenge (or can do so only with a low probability).

One of the main assumptions on which the security of distance bounding protocols relies is that the time that the prover spends in processing the verifier's challenge is negligible compared to the propagation time of the signal between the prover and the verifier. If the verifier overestimates the prover's processing time (i.e., the prover is able to process signals in a shorter time than expected), the prover will be able to pretend to be closer to the verifier. If the verifier underestimates this time (i.e., the prover needs more time to process the signals than expected), the computed distance bounds will be too large to be useful.

The challenge in implementing distance bounding protocols is therefore to implement a prover that is able to receive, process, and transmit signals in negligible time. This requirement can be easily met with ultrasonic distance bounding implementations where the prover's processing needs to be in the order of microseconds. However, because ultrasonic distance bounding is vulnerable to RF wormhole attacks, its use is limited to few specific applications. For most applications, radio distance bounding is the main viable way of verifying proximity to or a location of a device. In this case, the prover's processing time needs to be in the order of nanoseconds. Brands and Chaum in their original distance bounding work and Hancke and Kuhn in their follow-up work propose to use xor and comparison functions, respectively, to implement distance bounding at the prover. These functions require the prover to interpret the received bits before replying to the verifier, which results in long processing delays. Rasmussen and Capkun show that distance bounding can be implemented using analog processing functions that do not require the prover to interpret the received bits before replying to the verifier, resulting in sub-nanosecond processing delay at the prover.

Recommended Reading

1. Brands S, Chaum D (1993) Distance-bounding protocols (extended abstract) In: Proceedings of eurocrypt. Lecture notes in computer science, vol 765. Springer, pp 23–27
2. Capkun S, Hubaux JP (2006) Secure positioning in wireless networks. IEEE J Sel Area Commun: Special Issue on Security in Wireless Ad Hoc Networks 24(2):221–232
3. Hancke GP, Kuhn MG (2008) Attacks on time-of-flight distance bounding channels. In: ACM conference on wireless network security (WiSec), Alexandria, March 31–April, 2, 2008
4. Rasmussen KB, Capkun S (2008) Location privacy of distance bounding protocols. In: Proceedings of the 15th ACM conference on computer and communications security (Alexandria, 27–31 Oct 2008), CCS '08. ACM, New York, pp 149–160
5. Rasmussen KB, Capkun S (2010) Realization of RF distance bounding. In: Proceedings of the USENIX security symposium
6. Sastry N, Shankar U, Wagner D (2003) Secure verification of location claims. In: Proceedings of the 2nd ACM workshop on wireless security (San Diego, 19–19 Sep 2003), WiSe '03. ACM, New York, pp 1–10. doi:http://doi.acm.org/10.1145/941311.941313

Security of Group Communication in Wireless Mesh Networks

Cristina Nita-Rotaru[1], Reza Curtmola[2], Jing Dong[3]
[1]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA
[3]Department of Computer Science, Purdue University, West Lafayette, IN, USA

Synonyms

Secure wireless multicast

Related Concepts

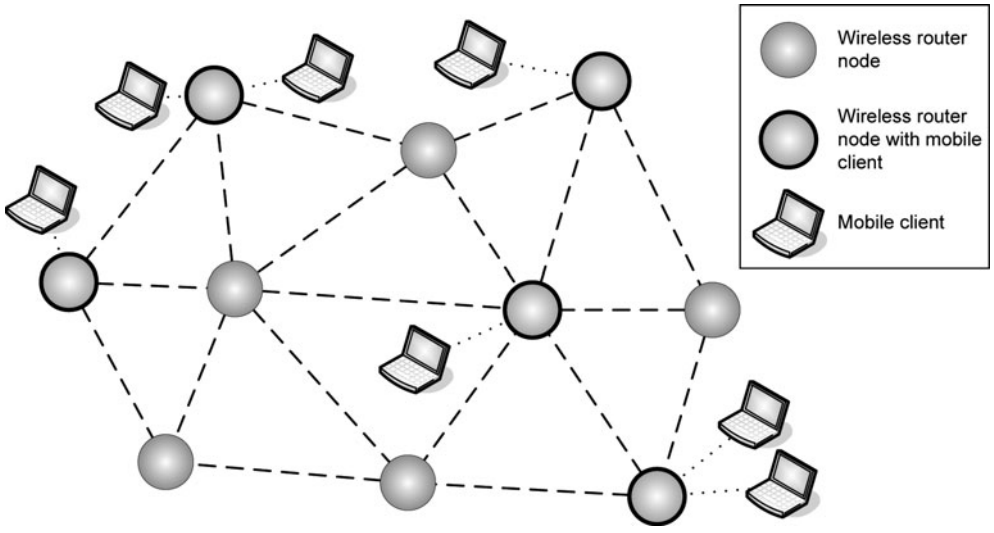
►Secure Network Protocols

Definition

Secure group communication for wireless mesh networks describes how to achieve confidentiality for one-to-many communication in wireless mesh networks in a scalable and efficient manner.

Background

Wireless mesh networks (WMNs) have emerged as a promising technology that offers low-cost high-bandwidth community wireless services. A WMN consists of a set of stationary wireless routers that form a multi-hop wireless backbone, and a set of mobile clients that communicate via the backbone routers (see Fig. 1). The community-oriented nature of WMNs facilitates group applications, such as webcast, distance learning, online gaming, video conferencing, and multimedia broadcasting.



Security of Group Communication in Wireless Mesh Networks. Fig. 1 Example mesh network

Many of these applications follow a group communication pattern in which one or more source clients disseminate data to a changing set of receivers. Communication is often achieved through scalable, high-throughput unicast and multicast wireless routing protocols. Some of these protocols were originally designed in the context of mobile ad hoc networks (MANETs) (e.g., AODV [9] DSR [4], and DSDV [8] for unicast and MAODV [10] for multicast). Others were specifically designed to take advantage of the mesh structure in a WMN (e.g., ODMRP [6]).

Security Goals for Group Communication. The openness of the wireless environment makes security a critical concern in deploying group applications in WMNs.

Many group-oriented applications require the dissemination of sensitive content. Examples include multimedia conferencing and applications which require only clients that have paid or registered for the service can receive data, such as online video broadcasting and distance learning. Thus, a major security goal for group applications is providing data confidentiality such that only current group members have access to the data sent to the group. More specifically, the goal is to provide the *group secrecy* property, such that it is computationally infeasible for a non-member node to discover the group data. This also includes the *backward* and *forward secrecy* properties which guarantee that it is computationally infeasible for a group member to gain access to the group data sent before the time it joins the group, or after the time it leaves (or is revoked from) the group, respectively. Many protocols assume that current group members do not leak keys to unauthorized parties.

Note that the underlying routing and MAC layer protocols can also be subjected to attacks. The major security goals for routing protocols is to achieve data delivery within certain performance parameters. For more details about vulnerability of routing protocols for wireless mesh networks and defense techniques against attacks, see Chap. ►Secure Routing in Wireless Mesh Networks.

Theory

Secure group communication is a mature research area and has a large body of research literature. Many protocols were proposed to achieve secure group communication in wired and wireless networks.

Approaches to Secure Group Communication

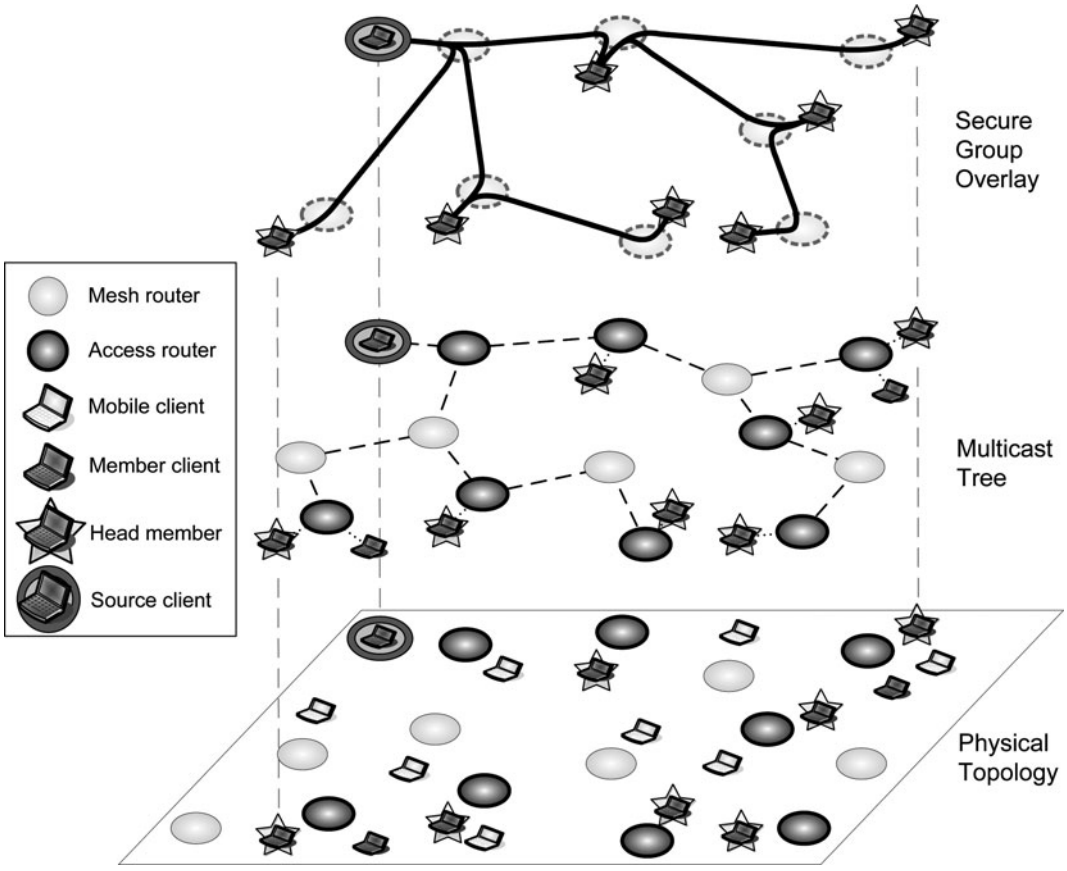
Secure group communication in wired networks. The most well-known protocols for single-source group multicast are centralized protocols like LKH [13] and its variants [7, 14, 15, 17, 18]. These protocols maintain global membership and require direct communication to the source node for both group joins and leaves. Reliable key delivery is achieved with redundant packets (forward error correction) and end-to-end unicast recovery. The high bandwidth and latency overhead of such protocols make them unsuitable for the multi-hop wireless environment. Recent work studied secure group communication in overlay multicast networks [1, 12, 16, 20]. Although overlay networks are also multi-hop in nature, they do not have the properties of client mobility, multiple clients sharing the same access router, or the much more limited bandwidth

resource as in WMNs. As a result, the protocols built for overlay networks are also not well suited for WMNs.

Secure group communication in wireless networks. Due to the constrained resource in the wireless environment, the basic approach to secure group communication in wireless networks is to match the protocol structure with the structure of wireless networks in order to achieve improved performance and efficiency. Below the authors briefly discuss existing protocols proposed for cellular networks and MANETs.

Cellular networks also exhibit a two-tier architecture, with high-bandwidth wired links connecting base stations (BSs) and low-bandwidth links between BS and clients. The work in [11] proposes a topology matching key management (TMKM) scheme that adapts the traditional key tree (LKH) to match the two-level structure of cellular networks. TMKM does not consider the problem of key distribution among BSs as they are connected with high-bandwidth wired links. However, in WMNs, mesh routers are connected with multi-hop wireless links, thus it is crucial to also consider the communication on the backbone routers in designing a secure group communication protocol.

The work in secure group communication in MANETs include GKMPAN [19], CRTDH [2], and the work in [5]. GKMPAN is a group key management scheme designed for a relatively stable group and focuses primarily on member revocation. The scheme uses pair-wise keys to distribute a common group key for data encryption among group members. CRTDH [2] relies on the Chinese Remainder Theorem and the Diffie–Hellman group key agreement for establishing group keys. It requires a number of messages linear to the group size to refresh the key for every group join and leave. Thus, it is not scalable in a wireless environment with limited bandwidth resource. Kaya et al [5] adopts a distributed scheme for handling group dynamics to address the high overhead of multi-hop communication in MANETs. However, the scheme fails to exploit the broadcast advantage of wireless communication, primarily due to the highly dynamic nature



Security of Group Communication in Wireless Mesh Networks. Fig. 2 Conceptual abstraction of the secure group overlay

of MANETs which prevents the establishment of a local subgroup key for data delivery.

Secure Group Overlay Based Group Communication in WMNs

The SeGrOM protocol proposed in [3] presents a secure group communication framework that is designed to address the specific characteristics of WMNs, such as the broadcast nature of wireless communication, the static nature of backbone routers, and the sharing of access routers among multiple clients.

In the high level, SeGrOM ensures group data confidentiality with respect to outsiders and routers by establishing a secure group overlay among member clients (see Fig. 2). The encrypted group data is delivered with a multicast tree connecting routers with member clients. Member join and leaves events are handled locally by their neighboring member clients.

As a router can have several clients connected to it, SeGrOM requires that a member client from each router is elected as the *head member* for the router. The head members of different routers form an overlay on top of the multicast tree by establishing a symmetric key with their upstream head member and each of their downstream head members using standard PKI techniques. This overlay is referred as the *secure group overlay* and the symmetric key as the *link key*. The join and leave of non-head members are handled by their local head member, while the join and leave of head members require only local link key reestablishments.

With the above setup, the flow of a group data packet is as follows. The source first encrypts the packet and forwards it to its access router, which then delivers the data packet to the head members in the network through the multicast tree. The head members participate in the secure data delivery protocol on the backbone network, and thus are able to decrypt the data received over the multicast tree. On receiving a data packet, the head members forward the data to the other members associated with the same router in a secure manner, e.g., by maintaining a common data encryption key in the local area.

A major component in this process is the selection of key used to provide data confidentiality among head members. SeGrOM considers two approaches. In the first approach, SeGrOM-Group, the data confidentiality is ensured with a common group key, which is distributed and refreshed using the secure overlay. In this case, the trade-offs are between global key refreshment overhead and partial loss of forward and backward secrecy. In the second approach, SeGrOM-Link, a unique data key is used for every data packet to ensure full forward and backward secrecy. SeGrOM also includes an optimization to SeGrOM-Link, SeGrOM-Hop, which reduces the computation and communication overhead in SeGrOM-Link with the help of hop keys.

SeGrOM-Group. In SeGrOM-Group, the data packet is encrypted with a common group key. The common group key is disseminated from the source on the secure group overlay. To avoid notifying the source for each group join and leave, a batching technique [7] is used, where the source refreshes the group key periodically irrespective of group dynamics. The advantage of this scheme is its improved performance and simplicity. However, as the key will not be changed immediately, when the group changes, nodes that have left the group will still be able to decrypt group data until the current key period elapses and a new group key is issued. This results in a partial loss of the forward and backward data secrecy due to the use of periodic group key refreshment.

SeGrOM-Link. In SeGrOM-Link, the source generates a random encryption key for each data packet. To allow group members to decrypt the data, this data encryption key is delivered hop-by-hop on the secure group overlay, alongside the data, protected by overlay link keys. Since link keys can be changed immediately upon changes in head members, SeGrOM-Link ensures full forward and backward data secrecy.

SeGrOM-Hop. SeGrOM-Hop optimizes the delivery of the data encryption keys in SeGrOM-Link by maintaining a *hop key*, shared between each head member and all of its downstream head members. With the hop key, the data encryption key must only be re-encrypted and broadcast once, instead of requiring a separate encryption and unicast for each downstream head member. Since the data encryption key has to be delivered for every data packet, the savings on computation and communication can be significant for applications with high data rates.

Applications

Examples of applications that can benefit from secure group communication in wireless mesh networks services are applications which disseminate sensitive content, such as multimedia conferencing, and applications which seek to ensure that only clients that have paid or registered for service can receive data, such as online video broadcasting and distance learning.

Recommended Reading

1. Abad C, Gupta I, Yurcik W (2005) Adding confidentiality to application-level multicast by leveraging the multicast overlay. In: Proceedings of ADSN in ICDCSW '05 (2005), Columbus
2. Balachandran R, Ramamurthy B, Zou X, Vinodchandran N (2005) CRTDH: an efficient key agreement scheme for secure

group communications in wireless ad hoc networks. In: Proceedings of IEEE ICC '05, Seoul
3. Dong J, Ackermann KE, Nita-Rotaru C (2009) Secure group communication in wireless mesh networks. Ad Hoc Networks (Elsevier), Special Issue: Privacy and Security in Wireless Sensor and Ad Hoc Networks 7(8):1563–1576
4. Johnson DB, Maltz DA, Broch J (2001) DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins CE (ed) Ad hoc networking, chapter 5. Addison-Wesley, New York, pp 139–172
5. Kaya T, Lin G, Noubir G, Yilmaz A (2003) Secure multicast groups on ad hoc networks. In: Proceedings of SASN '03, San Diego
6. Lee SJ, Su W, Gerla M (2002) On-demand multicast routing protocol in multihop wireless mobile networks. Mob Netw Appl 7(6):441–453
7. Li X, Yang Y, Gouda M, Lam S (2001) Batch rekeying for secure group communications. In: Proceedings of WWW '01 (2001)
8. Perkins CE, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM 1994, London
9. Perkins CE, Royer EM (2000) Ad hoc on-demand distance vector routing. In: Ad hoc networking. Addison-Wesley, New York
10. Royer E, Perkins C (2000) Multicast ad hoc on- demand distance vector (maodv) routing, Internet-draf, July 2000
11. Sun Y, Trappe W, Liu KJR (2004) A scalable multicast key management scheme for heterogeneous wireless networks. IEEE/ACM Trans Netw 12(4):653–666
12. Torres R, Sun X, Walters A, Nita-Rotaru C, Rao S (2007) Enabling confidentiality of data delivery in an overlay broadcasting system. In: Proceedings of the 26th IEEE INFOCOM 2007, Anchorage
13. Wong C, Gouda M, Lam S (2000) Secure group communications using key graphs. Trans Netw 8(1):16–30
14. Wong C, Lam S (2000) Keystone: A group key management service. In: Proceedings of ICT 2000, Acapulco
15. Yang Y, Li X, Zhang X, Lam S (2001) Reliable group rekeying: a performance analysis. In: SIGCOMM '01, San Deigo
16. Yiu WP, Chan SH (2004) SOT: secure overlay tree for application layer multicast. In: ICC '04, Paris
17. Zhang X, Lam S, Lee DY (2004) Group rekeying with limited unicast recovery. Comput Netw 44(6):855–870
18. Zhang X, Lam S, Lee DY, Yang Y (2003) Protocol design for scalable and reliable group rekeying. ToN 11, 6 (2003)
19. Zhu S, Setia S, Xu S, Jajodia S (2004) GKMPAN: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Mobiquitous '04, Boston
20. Zhu S, Yao C, Liu D, Setia S, Jajodia S (2005) Efficient security mechanisms for overlay multicast-based content distribution. In: Proceedings of Applied Cryptography and Network Security (ACNS) conference (2005), New York

Security of Web Browser Scripting Languages

►Script Language Security

Security of Wireless Mesh Networks (General Overview)

Reza Curtmola[1], Jing Dong[2], Cristina Nita-Rotaru[2]
[1]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA
[2]Department of Computer Science, Purdue University, West Lafayette, IN, USA

Synonyms

Secure networks design

Related Concepts

►Confidentiality; ►Key Management Protocols

Definition

Security of wireless mesh networks represents a collection of protocols and mechanisms designed to achieve security services such as authentication, integrity, and confidentiality for protocols used in the context of wireless mesh networks.

Background

Wireless mesh networks (WMNs) have emerged as a promising technology that offers low-cost high-bandwidth community wireless services. The self-maintenance feature of WMNs and the low cost of wireless routers make WMNs a promising technology for providing economic solutions for a wide range of applications, such as broadband community wireless service, enterprise wireless networking, security surveillance, and emergency networking [4]. In addition, through bridging and gateway functions of mesh routers, WMNs can be easily integrated with other networks, such as Internet, cellular networks, wireless local area networks (WLAN), wireless personal area networks (WPAN), wireless metropolitan area networks (WMAN), and sensor networks.

Given the large number of potential applications, WMNs have attracted tremendous interest from both academia and industry, and resulted in dedicated subworking groups in several industry standards groups, such as 802.11, 802.15, and 802.16 [10, 18, 29].

Group-oriented applications are an important class of applications on WMNs, given the community-oriented nature of WMN deployments. Example applications include webcast, distance learning, online gaming, video conferencing, and multimedia broadcasting. Many of these applications follow a communication pattern in which one or more source clients disseminate data to a changing set of receivers.

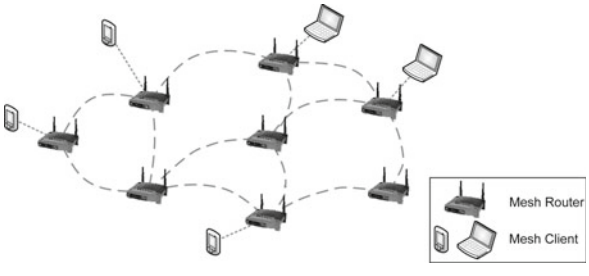
Wireless Mesh Networks

A wireless mesh network consists of a set of stationary wireless mesh routers and a set of wireless mesh clients (see Fig. 1). Each mesh router may be equipped with one or more wireless transceivers. As the radio range is insufficient to cover the entire deployment area, to communicate with each other, mesh routers form a multi-hop wireless network and relay traffic for each other.

Each mesh client is associated with a mesh router, and communicates with other mesh clients via the multi-hop network formed among the mesh routers. A mesh client may be mobile, and the state of the client is maintained with a hand-off protocol between mesh routers.

Communication in wireless mesh networks is done using wireless transmissions in a multi-hop fashion where the communication between source and destination is relayed by intermediate nodes. Routing protocols that were proposed in MANETs for unicast (e.g., AODV [25] DSR [19], DSDV [24]) and multicast (e.g., MAODV [29]) can be used in WMNs. In addition, some routing protocols such as ODMRP [22] were specifically designed to take advantage of the mesh structure in a WMN.

Driven by the increasing demand for rich and high-speed content access, recent research in WMNs has focused on developing high-performance communication protocols and given rise to two general design approaches, *dynamic topology-aware adaptation* and *dynamic network coding*. The main principle of dynamic topology-aware adaptation is to improve performance by dynamically adjusting protocol structure to accommodate the unstable nature of wireless links. Such adaptation is performed based on new routing metrics such as ETX [9], PP [14, 21], RTT [2], SPP [27] which capture the quality of the links and are better metrics for selecting high-throughput paths than the traditional hop count metric. Dynamic network coding, on the other hand, improves performance by exploiting the broadcast nature of wireless communication to effectively make use of the common occurrence of packet overhearing in wireless networks. The core principle of network coding is that intermediate nodes actively mix (or *code*) input packets and forward the resulting coded packets. Both approaches have been adopted in a wide range of practical systems and demonstrated significant improvement in system performance.



Security of Wireless Mesh Networks (General Overview).
Fig. 1 An example of a wireless mesh network

Theory

There are several aspects that make wireless networks in general and wireless mesh networks less robust and more vulnerable than their wired counterpart. High error rates, variable and unpredictable characteristics of the signal strength and propagation fluctuations with time and environment frequently result in broken links. Thus, wireless medium provides less robust communication than wired networks, making robustness a major challenge when designing group services operating on wireless networks. Scaling protocols is burdened by the shared nature of the medium, the interference and the limited bandwidth. Last but not least, security is more challenging in WMNs because the open wireless medium is more susceptible to attacks at all network levels and the multi-hop communication makes services more vulnerable to attacks coming from within the network. Specifically, as wireless communication is inherently open, attackers can easily set up rogue nodes to mount a wide variety of attacks, such as eavesdropping, jamming, man-in-the-middle, and spoofing. Software bugs, mis-configurations, and easy physical access all make mesh routers easier to be subverted by attackers and result in insider attacks.

Unfortunately, achieving security in wireless networks is much more challenging than in wired networks, especially for protocols aiming at high performance. On the one hand, wireless networks have a much more open communication environment and are subjected to a much wider range of attacks; on the other hand, wireless networks are much more limited in resource, thus only lightweight solutions can be applied. As a consequence, many security protocols proposed for wired networks are inapplicable in the wireless environment.

Based on the specific architecture in designing wireless mesh networks and the applications envisioned to be deployed on them, three aspects related to the security of wireless mesh networks are discussed: (1) group communication security for wireless mesh networks, (2) secure network coding, and (3) secure routing for dynamic-topology adaptation architectures.

Group communication security

Many of the applications envisioned for wireless mesh networks are group applications. A major security goal for group applications is providing data confidentiality such

that only current group members have access to the data sent to the group. Previous communication must remain protected from newly joined members, and future communication must be protected from members who have left the group.

Existing solutions for wired networks [1, 26, 30, 31, 34] are not well suited for WMNs as they assume efficient reliable end-to-end and multicast delivery, which is much more expensive to achieve in a multi-hop wireless environment. These protocols also do not exploit unique features of WMNs, such as the broadcast nature of wireless communication. Solutions proposed for wireless sensor networks (WSNs) [7, 15, 16] and MANETs [6, 20, 36] are designed to sustain severe computation power, storage, mobility, and energy constraints, and as a result have limited scalability and robustness.

Most of the secure group communication protocols for wireless networks were designed in the context of MANETs. Examples include GKMPAN [36], CRTDH [6], and the work in [20]. GKMPAN is a group key management scheme designed for a relatively stable group and focuses primarily on member revocation. The scheme uses pair-wise keys to distribute a common group key for data encryption among group members. CRTDH [6] relies on the Chinese Remainder Theorem and the Diffie–Hellman group key agreement for establishing group keys. It requires a number of messages linear to the group size to refresh the key for every group join and leave. Thus, it is not scalable in a wireless environment with limited bandwidth resource. Kaya et al. [20] adopts a distributed scheme for handling group dynamics to address the high overhead of multi-hop communication in MANETs. However, the scheme fails to exploit the broadcast advantage of wireless communication, primarily due to the highly dynamic nature of MANETs which prevents the establishment of a local subgroup key for data delivery.

SeGrOM [11] is a framework designed especially for wireless mesh networks. The framework employs decentralized group membership, promotes localized communication, and leverages the wireless broadcast nature to efficiently accommodate dynamic group changes and reduce communication overhead. The framework supports three secure multicast variants, with different tradeoffs in complexity, cost, and security, and an efficient group revocation protocol that exploits localized client mobility.

For more details about secure group communication in wireless mesh networks, refer article ►Security of Group Communication in Wireless Mesh Network.

Secure routing

A routing protocol in a WMN ensures that data sent by the source is routed across the network in a multi-hop fashion and reaches the destination. In this process, the routing is vulnerable to a wide variety of attacks because of the characteristics of the open wireless transmission medium and the lack of physical security of the mesh nodes. Attacks against routing range from simple attacks such as packet injection, modification, or replay (which can be performed by outside attackers) to more sophisticated attacks that are performed by inside attackers, such as packet dropping, flood rushing, and the wormhole attack. Given their static nature and their use for applications that require high-throughput, WMNs can also be subjected to metric manipulation attacks, which exploit the fact that routes are built based on the quality of wireless links as reported by the mesh nodes. The attacks can be particularly challenging to defend against when multiple attackers collude and coordinate their malicious actions.

To thwart outsider attacks, basic cryptographic primitives are usually sufficient. However, mitigating attacks executed by insiders requires more complex techniques, and cryptography alone is not sufficient. Examples include the monitoring of nodes by their neighbors [23], the use of end-to-end acknowledgments between destination and source [5], the use of accusation mechanisms [12], or the use of specialized hardware [17]. For more details about attacks and defenses against routing in wireless mesh networks, refer article ►Secure Routing in Wireless Mesh Network.

Secure network coding

Wireless mesh networks building on the paradigm on network coding make numerous optimizations and design choices to solve a plethora of practical aspects. Unfortunately, many of these design choices result in protocols that have numerous security vulnerabilities.

For example, the very nature of packet mixing makes network coding systems vulnerable to a severe security threat known as *pollution attacks*, in which attackers inject corrupted packets into the network. Although packet injection is not a new attack, its impact on network coding is devastating. This is because as long as there is one corrupted packet that an intermediate node uses during the coding process, all the coded packets forwarded by the node will be corrupted. The result is an epidemic propagation of corrupted packets, as further nodes code and forward more corrupted packets.

Current work relevant to the security of network coding has focused exclusively on the packet pollution attack. To prevent pollution attacks, intermediate nodes need to verify that each received coded packet is a valid combination of native packets from the source. As a result, traditional digital signature schemes cannot be used to defend against pollution attacks because the brute force approach in which the source generates and disseminates signatures

of all possible combinations of native packets has a prohibitive computation and communication cost, and thus it is not feasible. Several cryptographic approaches were proposed to combat this attack [8, 32, 35]. In such approaches, the source uses cryptographic techniques to create and send additional verification information that allows nodes to verify the validity of coded packets. Polluted packets can then be filtered out by intermediate nodes. The proposed schemes rely on techniques such as homomorphic hash functions or homomorphic digital signatures. These schemes have high computational overhead, as each verification requires a large number of modular exponentiations. In addition, they require the verification information (e.g., hashes or signatures) to be transmitted separately and reliably to all nodes in advance; this is difficult to achieve efficiently in wireless networks.

Several recent schemes avoid the cost of using heavyweight cryptographic primitives [3, 13, 33]. For more details about attacks and defenses in wireless mesh networks relying on network coding, ►Secure Network Coding for Wireless Mesh Networks.

Applications

Examples of applications that can benefit from secure wireless mesh networks services are applications which disseminate sensitive content, such as multimedia conferencing, and applications which seek to ensure that only clients that have paid or registered for service can receive data, such as online video broadcasting and distance learning. In addition, wireless networks using secure routing protocols that can defend not only against outside attacks but also against inside attacks are appropriate for deployment in environments with high security and availability requirements such as emergency deployments, natural disasters, military battle fields, and rescue missions.

Recommended Reading

1. Abad C, Gupta I (2005) Adding confidentiality to application-level multicast by leveraging the multicast overlay. In: Proceedings of ADSN 2005, Hilton
2. Adya A, Bahl P, Padhye J, Wolman A, Zhou L (2004) A multi-radio unification protocol for ieee 802.11 wireless networks. In: Proceedings of Broad Nets '04, Lausanne, Switzerland, pp 344–354
3. Agrawal S, Boneh D (2009) Homomorphic macs: mac-based integrity for network coding. In: Proceedings of ACNS '09, LNCS vol 5536. Springer, Heidelberg, pp 292–305
4. Akyildiz IF, Wang X, Wang W (2005) Wireless mesh networks: a survey. Comput Netw 47(4):445–487
5. Awerbuch B, Curtmola R, Holmer D, Nita-Rotaru C, Rubens H (2007) Odsbr: an on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks. ACM Transactions on Information and System Security (TISSEC), Bostn, 2007
6. Balachandran RK, Ramamurthy B, Xukai Z, Vinodchandran NV (2005) CRTDH: an efficient key agreement scheme for secure group communications in wireless ad hoc networks. In: Proceedings of ICC 2005, La Coruna
7. Chan H, Perrig A, Song D (2003) Random key predistribution schemes for sensor networks. In: Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy (S&P '03), Berkeley, pp 197–213
8. Charles D, Jain K, Lauter K (2006) Signatures for network coding. In: Proceedings of the conference on information sciences and systems (CISS '06), Princeton
9. Couto DSJD, Aguayo D, Bicket JC, Morris R (2003) A high-throughput path metric for multi-hop wireless routing. In: Proceedings of MOBICOM '03 (2003), ACM, San Diego
10. Djukic P, Valaee S (2007) 802.16 mesh networking. In: Ahson S, Ilyas M (eds) WiMax: standards and security, CRC Press, New York, pp 147–174
11. Dong J, Ackermann KE, Nita-Rotaru C (2009) Secure group communication in wireless mesh networks. Ad Hoc Networks (Elsevier), Special Issue: Privacy and Security in Wireless Sensor and Ad Hoc Networks 7, 2009
12. Dong J, Curtmola R, Nita-Rotaru C (2008) On the pitfalls of using high-throughput multicast metrics in adversarial wireless mesh networks. In: Proceedings of the 4th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08), California
13. Dong J, Curtmola R, Nita-Rotaru C (2009) Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In: Proceedings of WiSec 2009, ACM Press, Zurich
14. Draves R, Padhye J, Zill B (2004) Comparison of routing metrics for static multi-hop wireless networks. In: Proceedings of SIGCOMM '04, Portland, pp 133–144
15. Du W, Deng J, Han Y, Varshney P (2006) A key predistribution scheme for sensor networks using deployment knowledge. In: IEEE TDSC 3(1):62–77
16. Eschenauer L, Gligor VD (2002) A key-management scheme for distributed sensor networks. In: Proceedings of CCS '02, ACM Press, Washington
17. Hu L, Evans D (2004) Using directional antennas to prevent wormhole attacks. In: Proceedings of NDSS, San Diego
18. Ieee 802.15 standard group
19. Johnson DB, Maltz DA, Broch J (2001) DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins (ed) Ad hoc networking. Addison-Wesley, Reading, chapter 5, pp 139–172
20. Kaya T, Lin G, Noubir G, Yilmaz A (2003) Secure multicast groups on ad hoc networks. In: Proceedings of SASN '03, ACM, New York
21. Keshav S (1993) A control-theoretic approach to flow control. In: Proceedings of the Conference on Communications Architecture and Protocols, San Francisco, pp 3–15
22. Lee SJ, Su W, Gerla M (2002) On-demand multicast routing protocol in multihop wireless mobile networks. Mob Netw Appl 7(6):441–453
23. Marti S, Giuli T, Lai K, Baker M (2000) Mitigating routing misbehavior in mobile ad hoc networks. In: Proceedings of MOBICOM, Boston, 2000
24. Perkins CE, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM, London, 1994

25. Perkins CE, Royer EM (2000) Ad hoc On-Demand Distance Vector Routing. In: Perkins (ed) Ad hoc Networking. Addison-Wesley, Reading
26. Perrig A, Song D, Tygar JD (2001) ELK, a new protocol for efficient large-group key distribution. In: IEEE Symposium on Security and Privacy 2001, Oakland, pp 134–146
27. Roy S, Koutsonikolas D, Das S, Hu C (2006) High-throughput multicast routing metrics in wireless mesh networks. In: IEEE International Conference on Distributed Computing Systems (ICDCS), Lisboa, 2006
28. Royer E, Perkins C (2000) Multicast ad-hoc on-demand distance vector (MAODV) routing. In: Internet Draft (July 2000)
29. Status of project ieee 802.11s
30. Torres R, Sun X, Walters A, Nita-Rotaru C, Rao S (2007) Enabling confidentiality of data delivery in an overlay broadcasting system. In: INFOCOM 2007. Anchorage
31. Wong C, Gouda M, Lam S (2000) Secure group communications using key graphs. Trans Netw 8(1):16–30
32. Yu Z, Wei Y, Ramkumar B, Guan Y (2008) An efficient signature-based scheme for securing network coding against pollution attacks. In: Proceedings of INFOCOM '08, San Francisco
33. Yu Z, Wei Y, Ramkumar B, Guan Y (2009) An efficient signature scheme for securing XOR network coding against pollution attacks. In: Proceedings of IEEE INFOCOM, Rio de Janeiro, 2009
34. Zhang X, Lam S, Liu H (2005) Efficient group rekeying using application layer multicast. In: ICDCS '05, Columbus
35. Zhao F, Kalker T, Medard M, Han K (2007) Signatures for content distribution with network coding. In: Proceedings of ISIT '07, Nice
36. Zhu S, Setia S, Xu S, Jajodia S (2004) (Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Mobiquitous vol 00, pp 42–51

Security Reduction

Mehdi Tibouchi
Laboratoire d'informatique de l'ENS, École normale supérieure, Paris, France

Related Concepts

►Computational Assumption

Definition

A security reduction is a particular type of mathematical proof that some cryptographic primitive or protocol is secure, in the sense that it is "at least as difficult to break" as some other problem believed to be hard.

Background

It is usually not possible to prove that a practical cryptographic primitive is secure in an absolute, *informational* sense, that is, against an adversary with unlimited computational power. For example, an informationally secure encryption scheme must use a key at least as long as the message to be encrypted, and thus, has limited usefulness.

Real adversaries, however, do not have unlimited computational resources. This can be formalized in the language of complexity theory: the complexity of some problems is simply too high for them to be realistically tractable. That is, in particular, one of the insights underlying public key cryptography, as envisioned by Diffie and Hellman in the 1970s [5].

It seemed difficult to turn this insight into actual proofs of security for the protocols proposed at the time, such as RSA encryption or Diffie-Hellman key exchange, if only because the problems involved (factoring, discrete logarithm, etc.) are, to this day, only *conjectured* to be computationally intractable, and it is unlikely to see these conjectures settled soon failing deep, unforeseen advances in complexity theory.

Even *assuming* that factoring is hard, it is not known whether RSA encryption is secure (i.e., one-way). However, Rabin showed in 1979 [9] how to construct a public key encryption scheme in which recovering a message from a corresponding ciphertext is as hard as factoring the public key: this was possibly the first example of a security reduction.

Later, Goldwasser and Micali, in their 1983 paper *Probabilistic Encryption* [4], introduced a more formal and systematic approach to *provable security* [8], in which security notions themselves are defined in terms of probabilistic polynomial-time algorithms with oracles; they used this setting to define *semantic security* for public key encryption schemes, and construct the first example of a public key schemes secure in this stronger sense, provided that the quadratic residosity problem is hard. These notions are still in use today largely in the same form, and have found their way into international cryptographic standards, and symmetric cryptography as well.

Theory

The general structure of a security reduction can be described as follows. The aim is to prove that some cryptographic protocol C is secure in the sense that no probabilistic polynomial-time adversary can "break" it with nonnegligible probability. To do so, assume that such an adversary $\mathcal{A}$ exists, and construct another probabilistic polynomial-time algorithm $\mathcal{S}$ (the *simulator*), using $\mathcal{A}$ as an oracle, which solves a problem P that is believed to be hard. This is a proof by contradiction that C is secure under the assumption that P is hard.

As an example, one can show that breaking the one-wayness of Rabin encryption is as hard as factoring. Recall that Rabin encryption uses the product N of two large primes as a public key, and that a message $m \in \mathbb{Z}_N^*$ is

encrypted as $c = m^2 \bmod N$. The ciphertext c then decrypts as one of its four square roots mod N (this ambiguity can be lifted in a number of ways). An adversary $\mathcal{A}$ against the one-wayness of this scheme takes as input a public modulus N and a ciphertext c, and returns a decryption of c with nonnegligible probability ε.

Suppose that such an adversary $\mathcal{A}$ exists, and consider the following probabilistic polynomial-time simulator $\mathcal{S}$. $\mathcal{S}$ takes as input the product N of two large primes. It picks a random element $m_0 \in \mathbb{Z}_N^*$, computes $c = m_0^2 \bmod N$ and submits (N, c) to $\mathcal{A}$. With probability ε, $\mathcal{A}$ returns a square root m_1 of c mod N, and since c has four square roots, then $m_1 \not\equiv \pm m_0 \pmod N$ with probability 1/2. In that case, $p = \gcd(m_1 - m_0, N)$ is a nontrivial factor of N since $(m_1 - m_0)(m_1 + m_0) \equiv c - c \equiv 0 \pmod N$. Thus, $\mathcal{S}$ can return a nontrivial factor p of N with nonnegligible success probability $\varepsilon/2$. This concludes the proof that breaking the one-wayness of Rabin encryption is as hard as factoring.

A final remark on tightness. In this example, it is in fact only "as hard" up to the factor of 2 lost in the success probability. Since 2 is a small constant, this is considered a *tight* reduction: it is almost as easy to factor N as it is to recover a plaintext from the corresponding ciphertext. However, proofs of security can sometimes involve large polynomial losses in success probability ("loose reductions"), in which case the implications for the choice of security parameters are significant [7].

Applications

Early public key protocols with strong proofs of security (e.g., semantically secure encryption schemes, especially under chosen-ciphertext attacks) tended to be too inefficient for practical applications, and ad hoc substitutes such as PKCS#1 v1.5 [6] were used instead.

The situation started to change in the mid-1990s, however, in the wake of Bellare and Rogaway's work on the random oracle model [1]. Together with the availability of efficient cryptographic hash functions, it made efficient schemes with strong proofs of security possible [10] (assuming random oracles). Such schemes as RSA-OAEP [2] for encryption and RSA-PSS [3] for signature are now international standards and are gaining ever wider industry adoption.

Recommended Reading

1. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In ACM Conference on Computer and Communications Security, pp 62–73
2. Bellare M, Rogaway P (1994) Optimal asymmetric encryption. In EUROCRYPT, pp 92–111
3. Bellare M, Rogaway P (1996) The exact security of digital signatures – how to sign with rsa and rabin. In EUROCRYPT, pp 399–416
4. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comput Syst Sci 28(2):270–299
5. Hellman M, Diffie W (1976) New directions in cryptography. IEEE Trans Inf Theory 22:644–654
6. Kaliski B (1993) PKCS#1: RSA encryption standard, version 1.5. Technical report, RSA Laboratories
7. Koblitz N, Menezes A (2007) Another look at "provable security". J Cryptol 20(1):3–37
8. Pointcheval D (2005) Provable security for public key schemes, pp 133–189. Advanced Courses in Mathematics – CRM Barcelona. Birkhäuser Basel
9. Rabin MO (1979) Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science
10. Stern J (2003) Why provable security matters? In: Biham E (ed) EUROCRYPT. Lecture notes in computer science, vol 2656. Springer, pp 449–461

Security Standards Activities

Russ Housley
Vigil Security, LLC, Herndon, VA, USA

Definition

A security standard defines a set of features that provide one or more security service, including data integrity, authentication, non-repudiation, data confidentiality, and access control. Some security standards are embedded within a data protocol, others are defined separately for use with many different data protocols, and still others define infrastructure.

Background

This entry describes a number of highly visible security standards activities. It cannot be exhaustive, but it does include many standards bodies that are influencing the security industry and product development. Many of the standards are interrelated; for example, *X.509* public-key certificates have been profiled for use in the Internet by the PKIX working group of the Internet Engineering Task Force (IETF), and that profile has been augmented for Qualified Certificates, which are used to identify human beings involved in electronic commerce.

Activities

X.509

ITU-T Recommendation X.509 defines public-key *certificates* and *attribute certificates*. ITU-T, previously called CCITT, has been developing telecommunications

standards for decades. X.509 [1, 2] is part of a joint effort between ITU-T and the International Organization for Standardization, usually called ISO. The X.500 series of documents have numbers assigned by both standards bodies, but the numbers assigned by ITU-T are traditionally used to refer to the documents. Within this series, X.509 defines the public-key certificate to provide authentication in a ubiquitous global directory environment. While the envisioned directory deployment has never materialized, the certificate format has been used in small, closed networks as well as large, open deployments. Public-key certificates enable secure communication between entities that were unknown to each other prior to the communication. Deployments of public-key certificates are known as *public-key infrastructure* (PKI). To bring PKI to large multinational corporations or to millions of Internet users, a common certificate format is necessary. X.509 defines a flexible, extensible certificate format. The widespread adoption of X.509 is due to two factors. First, X.509 is technically suitable for many environments. Second, it was developed at an important time. It became an international standard at a time when many vendors were ready to implement certificate-based products.

X.509 includes a powerful extension mechanism. It was defined for Version 3 certificates and Version 2 CRLs (*Certificate Revocation Lists*). Arbitrary extensions can be defined and incorporated into a certificate or CRL, and a criticality flag indicates whether or not a certificate using system needs to properly handle this extension as part of the verification process. Certificate and CRL contents can readily be tailored to specific environments, and the inclusion of a particular critical extension can restrict use of the certificate to a particular application environment.

X.509 also specifies the attribute certificate format. An attribute certificate is used in conjunction with a public-key certificate to provide additional information, especially authorization information, about the named entity.

The ITU-T continues to maintain X.509 as well as develop X.509 enhancements. Most of the maintenance takes the form of clarifying text, and most of the enhancements take the forms of new standard extensions. Any problems found through operational experience are addressed in the standard through a formal defect reporting and resolution process.

PKIX

The Internet Engineering Task Force (IETF) is responsible for creating standards for the Internet. For the most part, the IETF develops protocols. This work is carried out by a number of working groups, which are organized into Areas. Within the Security Area, the *PKIX (Public-Key Infrastructure using X.509)* working group was formed at the end of 1995 to tailor the X.509 public-key certificate to the Internet environment. The working group set out to define an Internet PKI. It did not take long for the group to realize that defining an Internet PKI was more extensive than profiling the X.509 certificate. Thus, the PKIX charter was written to encompass four major activities:

1. X.509 certificate and certificate revocation list (CRL) profile
2. Certificate management protocols
3. Operational protocols
4. Certificate Policy (CP) and Certification Practice Statement (CPS) framework

The first activity was the original motivating task. The profile includes detailed specification of the mandatory, optional, critical, and noncritical extensions in a *PKIX-compliant* certificate or CRL. The profile was first published in January 1999 [3]. It was subsequently updated in April 2002 [4] and May 2008 [5]. The most recent update provides guidance on the use of international character sets [6]. In addition, the qualified certificate profile was developed in January 2001 [7], and it was updated in March 2004 [8]. Also, the attribute certificate profile was developed in April 2002 [9].

The second activity was to specify the protocols for management operations required in the Internet PKI, including certification of entities and their key pairs, *certificate revocation*, key backup and recovery (►Key Management), *Certification Authority* (CA) key rollover, and cross-certification. Two competing protocols were developed: CMP [10] and CMC [11]. CMP was updated in September 2005 [12], and CMC was updated in June 2008 [13].

The third activity, operational protocols, was to specify the protocols for day-to-day Internet PKI operation, such as certificate retrieval, CRL retrieval, and online certificate status checking. The results, to date, include several important specifications. It tells how to use FTP and HTTP to access repositories [14], and how to use HTTP to access a certificate store [15]. Others tell how to use an LDAPv2 directory as a repository [16, 17]. Another specification defines the Online Certificate Status Protocol (OCSP) [18]. And, another one defines the Server-Based Certificate Validation Protocol (SCVP).

Finally, the fourth activity, guidance to CP and CPS authors, provides topics and formats for these documents. The guidance was originally published in March 1999 [19]. The American Bar Association's Information Security Committee joined forces with the PKIX working group to improve it; an update was published in November 2003 [20].

PKIX has played an essential role in bringing PKI concepts to the Internet. These protocols and functions make a PKI possible in the diverse Internet environment. The flexibility, extensibility, and generality can satisfy greatly differing requirements. The PKIX work continues to evolve, and over time charter has been expanded to include supportive additional work items such as time-stamping protocols. The *Time-Stamp Protocol* (TSP) [21] was published in August 2001.

LDAP

The Lightweight Directory Access Protocol (LDAP) [22] was originally conceived as a simple to describe and simple to implement subset of the capability of the X.500 Directory Access Protocol (DAP). Over time, the subset of functions and features has expanded. Today, it is used as the access protocol for many repositories, some of which are based on X.500 directories, but many are not. As part of this evolution, the "lightweight" aspect of the protocol has diminished. Nevertheless, many vendors worldwide use LDAPv2 [23] and LDAPv3 [24]. The IETF LDAPext Working Group has been formed to specify useful extensions for LDAPv3, such as an authentication and access control mechanism.

Interoperability between PKI products from many different vendors in an LDAP environment is facilitated by the LDAPv2 schema [17] and the LDAPv3 schema [25]. These specifications describe how to locate certificate and CRL information in LDAP-compliant repositories.

S/MIME

In 1995, a consortium of industry vendors led by RSA Data Security, Inc. developed a companion security solution to the Multipurpose Internet Mail Extensions (MIME) specifications, which are the basis for any e-mail message that goes beyond simple text. For example, an e-mail message that includes more than one font, bold text, or an attachment makes use of MIME. Secure MIME (S/MIME) specifies encryption and digital signatures for MIME messages. While a formal standards body did not develop the original S/MIME specifications, many important product vendors embraced S/MIME. To build on and expand this success, the consortium released change control of the S/MIME Version 2 documents [26, 27] to the IETF in 1997.

The IETF S/MIME Working Group developed significant enhancements, resulting in S/MIME Version 3 [28–32]. The primary focus of the S/MIME Working Group was to develop an algorithm-independent protocol and incorporate a number of new security features into the specifications, while preserving compatibility with the earlier specification whenever possible. In particular, the S/MIME Version 3 specifications include support for sending encrypted messages to large mail lists, security labels on messages (e.g., "company proprietary," "secret," or "top secret"), and signed message receipts. These signed receipts provide proof that the intended recipient received a signed message that contained a request for a receipt.

The S/MIME Version 3 specifications include discussion of PKI concepts such as certificate format, certificate processing, and CRLs. These specifications are compatible with the X.509 profile developed by the IETF PKIX Working Group, and they provide additional details for the use of X.509 certificates in the e-mail environment. Further, provision is made in the message envelope to carry an arbitrary numbers of certificates and CRLs to assist the recipient with certification path construction and validation.

The S/MIME Version 3.1 [33–36] specifications update the mandatory to implement cryptographic algorithms, and they include features to facilitate the migration from one algorithm suite to another.

IPsec

IPsec is designed to provide interoperable, high-quality, cryptographically based security – the Internet Protocol (both Version 4 (IPv4) and Version 6 (IPv6)). The security services offered include access control, connectionless integrity, data origin authentication, protection against replays, confidentiality, and limited traffic flow confidentiality. The services are provided by cryptographic key management procedures and protocols in conjunction with traffic security protocols. IPsec includes two traffic security protocols: the Authentication Header (AH) [37] and the Encapsulating Security Payload (ESP) [38].

Either the Internet Key Exchange (IKE) protocol [39] or IKEv2 [40] is used to establish the symmetric keying material needed by AH and ESP. IKE and IKEv2 provide for strong authentication of the IP layer entities based on many different techniques, including X.509-certificate-based authentication.

TLS

Netscape developed the Secure Sockets Layer (SSL). SSL Version 3.0 (SSLv3.0) was widely deployed to secure World Wide Web transactions, and in 1996, Netscape released change control for the protocol to the IETF. The *Transport Layer Security (TLS)* specification [41, 42] is the IETF standards-track version of SSLv3.0. This history has many parallels to S/MIME. The original specification was developed outside of any standards body, and then it was released to the IETF, who took over configuration control and made enhancements.

TLS creates a secure channel between two transport layer entities, providing certificate-based authentication, information integrity, and data confidentiality. TLS is usually used with an X.509 certificate to authenticate the server; however, while it is rarely used, a certificate can also be used to authenticate the client.

TLS provides security for connection-oriented protocols such as TCP. The Datagram TLS (DTLS) [43] was developed to provide similar security services to connectionless protocols such as UDP. DTLS has become an important security protocol in the Voice over IP (VoIP) application environment.

Secure Shell

Secure Shell (SSH) is a client-server protocol that is typically used to log into a remote computer and execute commands, but it also supports tunneling of other protocols such as X-windows.

In 1995, Tatu Ylönen, a researcher at Helsinki University of Technology in Finland, designed the first version of the SSH protocol. The goal of SSH was to replace the rlogin, TELNET, and rsh protocols, with one that prevented password-sniffing attacks.

In 1996, SSH version 2 was designed, which was incompatible with the original version, but offered improved security and the ability to run any number of shell sessions over a single SSH connection. This version of SSH was proprietary.

In 1999, developers that want a freely available version began the Open SSH effort.

In 2006, the IETF SECSH working group published SSH as a proposed Internet standard [44].

In 2009, the IETF ISMS working group, in an effort to provide security for network management, defined the way to protect SNMP (Simple Network Management Protocol) traffic with SSH [45].

AAA

In 1997, the IETF created the first standard Authentication, Authorization, and Accounting (AAA) protocol, called RADIUS (Remote Access Dial-In User Service) [46–48]. As the name implies, RADIUS is designed for use with Dial-In Access Servers. RADIUS has been a big success, displacing many proprietary protocols. RADIUS is widely implemented as Network Access Servers (NASs) serving analog and digital dial-in Point-to-Point Protocol (PPP) service, and it is the prevalent Internet Service Provider (ISP) access authentication protocol. RADIUS has been adapted for use with DSL (using PPPOE), cable access, and wireless local and wide area networks. RADIUS has been successful because it offers a simple and flexible model for client-server exchanges. However, this simple model does not have sufficient security for some new applications, and it also lacks support for server-initiated control.

The IETF AAA Working Group is responsible for building a more secure and capable AAA protocol. A number of proposals were evaluated in June 2000, and the working group selected the Diameter protocol [49]. Diameter is designed to be compatible with RADIUS to a great extent, but many of the messaging underpinnings have been upgraded to be more secure. Security is provided using CMS and IPsec. For better response time, the SCTP (Stream Control Transmission Protocol) transport is supported as an alternative.

Diameter explicitly supports server-to-client requests and message forwarding. These capabilities have previously been forced into RADIUS [50]. Diameter also includes explicit support for application suite additions. Application designs have been drafted for Mobile IP authentication and third-generation wireless telecommunications [51] sessions.

OpenPGP

As with the S/MIME and TLS Working Groups, the IETF OpenPGP Working Group was formed to develop a standard based on a protocol that was developed outside of any standards body. The popular *Pretty Good Privacy* (PGP) e-mail security package was brought to the IETF so that interoperable implementations from different vendors could be developed. OpenPGP [52] defines e-mail message protection and the PGP certificate format (an alternative to X.509). Despite a loyal installed base, OpenPGP has not seen great adoption in many corporate or government environments. Many view OpenPGP as an individual-to-individual solution. The user-centric trust model cannot easily be centrally controlled by an organization.

XML Security

Prominent standards bodies are actively developing XML (eXtensible Markup Language) security specifications, including the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS).

The W3C developed specifications for the XML syntax with respect to encryption [53] and digital signature [54]. In addition, W3C developed the XML protocols for key management (XML Key Management Specification, or XKMS) that allow a client to obtain key information (including values, certificates, management, or trust data) from a Web service. The XML Key Management Specification (XKMS) is made up of two parts: the XML Key

Information Service Specification (X-KISS) and the XML Key Registration Service Specification (X-KRSS).

The OASIS Security Services Technical Committee developed the Security Assertion Markup Language (SAML) [55]. SAML provides an XML framework for exchanging authentication and authorization information. The underlying authentication mechanism may be PKI-based, but SAML encompasses a number of other authentication technologies as well. A number of other OASIS technical committees and other standards bodies are making use of SAML. For example, SAML is used in Business Transaction Processing (BTP), electronic business XML (ebXML), Provisioning Services Markup Language (PSML), eXtensible Access Control Markup Language (XACML), Web Services Security (WSS), and Digital Signature Services (DSS).

KMIP

The OASIS Key Management Interoperability Protocol (KMIP) Technical Committee develops specifications to manage symmetric and asymmetric cryptographic keys. The Technical Committee will produce specifications for a client–server protocol, use cases, and usage guide. The protocol supports a number of operations such as registering existing keys, creating new keys, destroying unneeded keys, and archiving keys. The operations allow cryptographic objects to be managed in each of the cryptographic object states in NIST Special Publication 800-57. None of the specifications have been released for public comment yet.

IEEE P802

Local Area Network (LAN) and Metropolitan Area Network (MAN) standards encompass a number of data communications technologies and the applications of these technologies. There is no single technology that is applicable to all applications. Correspondingly, no single local or metropolitan area network standard is adequate for all applications. As a result, the Institute of Electrical and Electronics Engineers (IEEE) Standards Association sponsors several working groups and technical advisory groups within Project 802.

Security was the focus of IEEE 802.10, but the standards that it produced never saw much adoption. As a result, this series of standards has been withdrawn. Despite this situation, a recognized need for encryption of LAN and MAN traffic resulted in the development of an alternate solution. IEEE 802.1AE accommodates very-high-speed LAN environments.

IEEE 802.1X specifies port-based access controls. It provides a means of authenticating and authorizing devices attached to a LAN port, preventing access when authentication and authorization fails.

IEEE 802.11 includes the ability to encrypt wireless LAN traffic using the Wired Equivalent Privacy (WEP) protocol. Unfortunately, WEP has many flaws. IEEE 802.11i provides a short-term and a long-term replacement for WEP, called TKIP and CCMP, respectively. The Temporal Key Integrity Protocol (TKIP) is intended to replace WEP on hardware that was originally designed for WEP. TKIP is implemented by firmware and driver software upgrades. The Counter and CBC-MAC Protocol (CCMP) embraced significant security improvements, but CCMP is not compatible with the older hardware. Today, products implement both TKIP and CCMP, but TKIP is provided for compatibility with fielded devices.

IEEE 802.15 developed security solutions for personal area networks. Clearly, more customers are demanding security solutions. Other working groups are likely to have security initiatives in the future.

IEEE P1363

IEEE Project 1363 develops standard specifications for public-key cryptography, which includes mathematical primitives for key derivation, public-key encryption, and digital signatures. P1363 and P1363a have been adopted as IEEE standards. Work continues on documents that specify additional techniques.

ANSI X9F

The American National Standards Institute (ANSI) committee X9 (Financial Services) develops and publishes standards for the financial services industry. X9 also holds the U.S. vote for international banking standards in ISO Technical Committee 68. X9 standards facilitate delivery of financial products and services. X9F, the Data and Information Security Subcommittee, deals with standardization to reduce financial data security risk and vulnerability. X9F includes working groups that focus on cryptographic tools (X9F1), security protocols (X9F3), among others. X9F has published many standards (the X9 online catalog can be found at http://www.x9.org), and many of its standards become international standards through a close working relationship with ISO TC68.

FIPS

The U.S. National Institute of Standards and Technology (NIST) is responsible for the development of standards that must be followed by the U.S. Federal Government; they are called Federal Information Processing Standards (FIPS). These standards are followed by vendors that want to sell

products to the U.S. Federal Government, and they are followed by many others too.

FIPS Publications are issued by NIST after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Reform Act of 1996 and the Federal Information Security Management Act of 2002.

FIPS Publications cover a wide range of topics. The Advanced Encryption Standard (AES) is published in FIPS 197 [56]. AES is one example of a FIPS that has seen worldwide acceptance and deployment.

Influential Activities

Some activities that are not part of any formal security standards body are influencing security standards development, the security industry, and product development. Again, this discussion cannot be exhaustive, but a number of the highly visible security standards influencing activities are discussed.

U.S. FPKI

The U.S. Federal Public-Key Infrastructure (FPKI) is an initiative by the U.S. Government to define a PKI suitable for its own use. The FPKI include the Federal Bridge CA. Many PKI service providers want to cross-certify with the Federal Bridge so that users in the U.S. Federal Government are able to validate their certificates. Cross-certification requires a review of the vendor's certification policy using in the format specified by the IETF PKIX working group.

The U.S. FPKI also imposes requirements on vendors wanting to sell PKI products to the U.S. Government. To the greatest extent possible, commercial standards have been referenced and profiled. So far, the FPKI is sufficiently similar to other PKIs that meeting these requirements has not unduly restricted vendors.

JCP

The Java Community Process (JCP) is an open organization of international Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits. This group publishes Java Specification Requests (JSRs), and several are related to security and PKI. For example, JSR 55 discusses certification path creation, building, and verification; JSR 74 discusses many of the Public-Key Cryptography Standards (PKCS) published by RSA Laboratories; JSR 104 discusses XML trust services; JSR 105 discusses XML Digital Signature services; JSR 106 discusses XML Digital Encryption services; and JSR 155 discusses Web Services Security Assertions based on the OASIS SAML specification. Further information can be found at [57].

Recommended Reading

1. ITU-T (1997) Recommendation X.509: The Directory—Authentication Framework
2. ITU-T (2000) Recommendation X.509: The Directory—Public Key and Attribute Certificate Frameworks
3. Housley R, Ford W, Polk W, Solo D (1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459
4. Housley R, Polk W, Ford W, Solo D (2002) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280
5. Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W (2008) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280
6. Yergeau F (1998) UTF-8, a Transformation Format of ISO 10646. RFC 2279
7. Santesson S, Polk W, Barzin P, Nystrom M (2001) Internet X.509 Public Key Infrastructure Qualified Certificates Profile. RFC 3039
8. Santesson S, Nystrom M, Polk W (2004) Internet X.509 Public Key Infrastructure Qualified Certificates Profile. RFC 3739
9. Farrell S, Housley R (2002) An Internet Attribute Certificate Profile for Authorization. RFC 3281
10. Adams C, Farrell S (1999) Internet X.509. Public Key Infrastructure Certificate Management Protocols. RFC 2510
11. Myers M, Liu X, Schaad J, Weinstein J (2000) Certificate Management Messages over CMS. RFC 2797
12. Adams C, Farrell S, Kause T, Mononen T (2005) Internet X.509. Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210
13. Schaad J, Myers M (2008) Certificate Management Messages over CMS (CMC). RFC 5272
14. Housley R, Hoffman P (1999) Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP. RFC 2585
15. Gutmann P (2006) Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP. RFC 4387
16. Boeyen S, Howes T, Richard P (1999) Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv2. RFC 2559
17. Boeyen S, Howes T, Richard P (1999) Internet X.509 Public Key Infrastructure LDAPv2 Schema. RFC 2587
18. Myers M, Ankney R, Malpani A, Galperin S, Adams C (1999) X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP. RFC 2560
19. Chokhani S, Ford W (1999) Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 2527
20. Chokhani SW. Ford, Sabett R, Merrill C, Wu S (2003) Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 3647
21. Adams C, Cain P, Pinkas D, Zuccherato R (2001) Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161
22. Howes T, Smith M (1997) LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol. Macmillan Technical Publishing, Indianapolis
23. Yeong W, Howes T, Kille S (1995) Lightweight Directory Access Protocol. RFC 1777

24. Wahl M, Howes T, Kille S (1997) Lightweight Directory Access Protocol (v3). RFC 2251
25. Zeilenga K (2006) Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates. RFC 4523
26. Dusse S, Hoffman P, Ramsdell B, Lundblade L, Repka L (1998) S/MIME Version 2 Message Specification. RFC 2311
27. Dusse S, Hoffman P, Ramsdell B, Weinstein J (1998) S/MIME Version 2 Certificate Handling. RFC 2312
28. Housley R (1999) Cryptographic Message Syntax. RFC 2630
29. Rescorla E (1999) Diffie-Hellman Key Agreement Method. RFC 2631
30. Ramsdell B (ed) (1999) S/MIME Version 3 Certificate Handling. RFC 2632
31. Ramsdell B (ed) (1999) S/MIME Version 3 Message Specification. RFC 2633
32. Hoffman P (ed) (1999) Enhanced Security Services for S/MIME. RFC 2634
33. Housley R (2004) Cryptographic Message Syntax. RFC 3852
34. Housley R (2007) Cryptographic Message Syntax (CMS) Multiple Signer Clarification. RFC 4853
35. Ramsdell B (ed) (1999) S/MIME Version 3.1 Certificate Handling. RFC 3850
36. Ramsdell B (ed) (1999) S/MIME Version 3.1 Message Specification. RFC 3851
37. Kent S (2005) IP Authentication Header. RFC 4302
38. Kent S (2005) IP Encapsulating Security Payload (ESP). RFC 4303
39. Harkins D, Carrel D (1998) The Internet Key Exchange (IKE). RFC 2409
40. Kaufman C (ed) (2005) Internet Key Exchange (IKEv2) Protocol. RFC 4306
41. Dierks T, Allen C (1999) The TLS Protocol Version 1.0. RFC 2246
42. Dierks T, Rescorla E (2006) The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346
43. Rescorla E, NModadugu N (2006) Datagram Transport Layer Security. RFC 4347
44. Ylonen T, Lonvick C (ed) (2006) The Secure Shell (SSH) Protocol architecture. RFC 4251
45. Harrington D, Salowey J, Hardaker W (2009) Secure Shell Transport Model for the Simple Network Management Protocol (SNMP). RFC 5592
46. Rigney C, Willens S, Rubens A, Simpson W (2000) Remote Authentication Dial in User Service (RADIUS). RFC 2865
47. Rigney C (2000) RADIUS Accounting. RFC 2866
48. Rigney C, Willats W, Calhoun P (2000) RADIUS Extensions. RFC 2869
49. Mitton D, St.Johns M, Barkley S, Nelson D, Patil B, Stevens M, Wolff B (2001) Authentication, Authorization, and Accounting: Protocol Evaluation. RFC 3127
50. Mitton D (2000) Network Access Servers Requirements: Extended RADIUS Practices. RFC 2882
51. http://www.3gpp2.org/
52. Callas J, Donnerhacke L, Finney H, Shaw D, Thayer R (2007) OpenPGP Message Format. RFC 4880
53. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
54. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
55. http://www.oasis-open.org/specs/#samlv2.0
56. National Institute of Standards and Technology (2001) FIPS Pub 197: Advanced Encryption Standard (AES). 26 November 2001.
57. http://www.jcp.org/
58. http://ice-car.darmstadt.gmd.de/ice-car-home.html
59. Freeman T, Housley R, Malpani A, Cooper D, Polk W (2007) Server-Based Certificate Validation Protocol (SCVP). RFC 5055
60. http://www.w3.org/TR/2001/NOTE-xkms-20010330/

Security Token

►Authentication Token

Security Verification

►Formal Methods for the Orange Book

Segregation of Duties

►Separation of Duties

Selective Forgery

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery; ►Security Definition

Definition

Existential forgery is a certain type of attacker goal that is used to formally define the security of digital signature schemes, in particular the unforgeability part of security.

Theory

Selective forgery is a message-related ►forgery against a cryptographic ►digital signature scheme. Given a victim's verifying key, a selective forgery is achieved if the attacker finds a signature s for a message m selected by the attacker prior to the attack, such that the signature s is valid for m with respect to the victim's verifying key. Selective forgery defines the outcome of an attack, NOT the way how or how often the attacker can interact with the attacked signer while the attack is performed (►forgery and [1]). Important types of attacker goals used to formally define the security of digital signature schemes are in order

of increasing strength: existential forgery, selective forgery, and total break.

Recommended Reading

1. Menezes AJ, van Oorschoot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, p 432

Self-Shrinking Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Clock-Controlled Generator; ►Linear Feedback Shift Register; ►Shrinking Generator; ►Stream Cipher

Definition

The self-shrinking generator is a ►clock-controlled generator that has been proposed in [1]; it is strongly related to the ►shrinking generator, but uses only one ►Linear Feedback Shift Register (LFSR) R, producing a ►maximum-length sequence.

Background

Its principle is really easy to get: the output sequence of the LFSR is partitioned into pairs of bits. According to the value of the pair, one bit is added to the keystream, and then the pair is discarded and we go to the next one. More precisely:

Pair	Bit added
10	0
11	1
01	No bit added
00	No bit added

Example

Let us consider that R has length four, and that its feedback is given by $s_{t+1} = s_t + s_{t-3}$. If the initial state is $s_3 s_2 s_1 s_0 = 0101$, then the output of the LFSR is 10 10 11 00 10 00 11 11 01 01 10 01 00 01 11 10 10 11 00 10 00 11 11 ... This gives the following output for the whole scheme: 00101101001011....

Theory

A survey on the possible attacks has been provided in 2001 in [2], and more recent attacks have been published in [3–5]. The attack presented in [4] retrieves the initial state of the LFSR with a time complexity of $O(2^{0.556n})$ (respectively $O(2^{0.571n})$), a memory complexity of $O(n^2)$ from $O(2^{0.161n})$ (respectively $O(2^{0.194n})$) bits of keystream for $n \geq 100$ (respectively $n < 100$). This attack, as the one presented in [3], is independent of the Hamming weight of the feedback polynomial of the LFSR. In a more recent paper, the authors of [5] showed that the attack efficiency can be improved for some values of the Hamming weight of the feedback polynomial.

Some variants of the Self-Shrinking Generator have been proposed: [6], which cannot be considered more secure than the original one [7, 8]; and [9], whose security seems to be better for the moment.

Recommended Reading

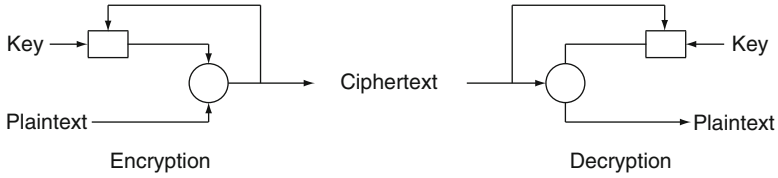
1. Meier W, Staffelbach O (1995) The self-shrinking generator. In: Advances in cryptology – Eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 205–214
2. Zenner E, Krause M, Lucks S (2001) Improved cryptanalysis of the self-shrinking generator. In: ACIPS'2001. Lecture notes in computer science, vol 2119. Springer, Berlin, pp 21–35
3. Hell M, Johansson T (2006) Two new attacks on the self-shrinking generator. IEEE Trans Info Theory 52(8):3837–3843
4. Zhang B, Feng D (2006) New guess-and-determine attack on the self-shrinking generator. In: ASIACRYPT'2006. Lecture notes in computer science, vol 4284. Springer, Berlin, pp 54–68
5. Debraize B, Goubin L (2008) Guess-and-determine algebraic attack on the self-shrinking generator. In: FSE'2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 235–252
6. Hu Y, Xiao G (2004) Generalized self-shrinking generator. IEEE Trans Info Theory 50(4):714–719
7. Zhang B, Wu H, Feng D, Bao F (2004) Security analysis of the generalized self-shrinking generator. In: Information and communications security: ICISC'04. Lecture notes in computer science, vol 3269. Springer, Berlin, pp 388–400
8. Dong L, Hu Y (2007) Weak generalized self-shrinking generators. J Syst Eng Electron 18(2):407–411
9. Kanso A (2010) Modified self-shrinking generator. Comput Eletr Eng 36:993–1001

Self-Synchronizing Stream Cipher

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest cedex 3, France

Synonyms

Asynchronous stream cipher



Self-Synchronizing Stream Cipher. Fig. 1 Self-synchronizing stream cipher

Related Concepts

►Stream Cipher

Definition

In a self-synchronizing, or asynchronous, stream cipher, the keystream depends on the secret key of the scheme, but also of a fixed number, say t, of ciphertext digits (that have already been produced, or read; this distinguishes it from a ►synchronous stream cipher).

Background

The idea of self-synchronization was patented in 1946 and has the advantage that the receiver will automatically synchronize with the keystream generator after receiving t ciphertext digits.

Example

A simple example of a self-synchronizing stream cipher is a block cipher in cipher-feedback mode (CFB).

Theory

A self-synchronizing stream cipher can be viewed as depicted in Fig. 1.

According to its design, such a scheme is able to resynchronize the keystream with the message with just a few correct bits of ciphertext. This means that if some bits are inserted or deleted in the ciphertext, just a small part of the plaintext will not be obtained correctly; the next set of t consecutive correct bits in the ciphertext will be sufficient to resynchronize the keystream and produce the following bits of the plaintext correctly.

But, if one bit of the ciphertext has been altered during the transmission, some errors will occur during the decryption of the next t bits; after this, decryption will go on correctly.

What can do an active attacker with such a scheme? According to the propagation of each error in a ciphertext on about t bits of plaintext, it is more difficult for an attacker to forge a plaintext of its choice than in a *synchronous stream cipher*. Moreover, it is also more difficult for him to desynchronize the keystream since the scheme is able to resynchronize it by itself. If the attacker wants to desynchronize all the keystream, he has to do a lot of modifications on the ciphertext [1]. Nevertheless, some complementary mechanisms, that can ensure authentication or integrity of the ciphertext are welcome to help the receiver checking that all is going well.

At last, since each plaintext digit influences the whole ciphertext (through the feedback of the ciphertext on the keystream generation), the statistical properties of the plaintext are dispersed in the ciphertext, and such a scheme may be more resistant against attacks based on plaintext redundancy than *synchronous stream ciphers*.

The security of self-synchronizing stream ciphers is strongly related to the nonlinearity of Boolean functions, as discussed in [2–4].

Applications

Designs of self synchronizing stream ciphers have been proposed in [2, 3], and more recently in [5].

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers. Springer, Berlin
2. Maurer UM (1991) New approaches to the design of self-synchronizing stream ciphers. Advances in Cryptology – Eurocrypt'91. Lecture notes in computer science, vol 547. Springer, pp 458–471
3. Daemen J, Govaerts R, Vandewalle J (1992) On the design of high speed self-synchronizing stream ciphers. Singapore ICCS-ISITA'92 conference proceedings, pp 279–283
4. Guillot P, Mesnager S (2005) Non-linearity and security of self synchronizing stream ciphers. 2005 International symposium on nonlinear theory and its applications, Belgium, 18–21 Oct 2005
5. Daemen J, Kitsos P (2008) The self-synchronizing stream cipher moustique, new stream cipher designs. Lecture notes in computer science, vol 4986. Springer, pp 210–223

Semantic Security

Kazue Sako
NEC, Kawasaki, Japan

Synonyms

Indistinguishability of encryptions

Related Concepts

►Public Key Cryptosystem; ►Non-Malleability

Definition

Semantic security is a notion to describe the security of an encryption scheme.

An adversary is allowed to choose between two plaintexts, m_0 and m_1, and he receives an encryption of either one of the plaintexts. An encryption scheme is semantically secure, if an adversary cannot guess with better probability than 1/2 whether the given ciphertext is an encryption of message m_0 or m_1. The notion is also referred to as *indistinguishability of encryptions* and noted as IND. Historically, the word "semantic" came from the definition that the encryption reveals no information no matter what kind of semantics are embedded in the encryption. It has been proven that the definition describing this requirement is equivalent to the indistinguishability of encryptions.

Background: The notion of semantic security was developed in order to treat rigid security of encryption schemes.

Theory

The notion of semantic security can be further distinguished by the power of adversary. More specifically, a powerful adversary may have access to an encryption *oracle* and/or decryption oracle at various stages of the guessing game. Here, an encryption oracle is an oracle that provides an encryption of a queried plaintext, and a decryption oracle provides the decryption of a queried ciphertext.

The notion of semantic security can be applied to both ►symmetric cryptosystems and ►Public Key Cryptosystems. But since the concrete security analysis of a public key encryption scheme is more tractable, the term is more frequently used to discuss the security of public key encryption schemes.

In a public key encryption scheme, the adversary can always access the encryption oracle, because he can encrypt by himself. Therefore the semantic security must be achieved against such an adversary. Such security is called "semantically secure against ►chosen plaintext attack" and written IND-CPA. The threat of adversary who has access to decryption oracle is called ►chosen ciphertext attack (CCA). If a public-key scheme is semantically secure against an adversary who has access to a decryption oracle before determining the pair of plaintexts m_0 and m_1, it is called IND–CCA1. If a public-key scheme is semantically secure against an adversary who has access to a decryption oracle not only before receiving a target ciphertext but also during the guessing stage, then it is defined as IND–CCA2. It is regarded that this type of adversary is the most powerful. Therefore the scheme achieving IND–CCA2 is considered most secure. (There is a restriction on this type of adversary, namely that he cannot receive an answer of the target ciphertext from decryption oracle.) Besides semantic security, there are related notions such as ►Non-Malleability and Plaintext Awareness.

Recommended Reading

1. Bellare MA, Desai D Pointcheval, Rogaway P (1998) Relations among notions of security for public-key encryption schemes. In: Krwawczyk H (ed) Advances in cryptography—CRYPTO'98, Lecture notes in computer science, vol 1462. Springer, Berlin, pp 26–45

Sender Anonymity

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Definition

Sender ►anonymity is achieved in a messaging system if an eavesdropper who picks up messages from the communication line of a recipient cannot tell with better probability than pure guessing who sent the messages. During the attack, the ►eavesdropper may also listen on all communication lines of the network including those that connect the potential senders to the network and he may send his own messages. It is clear that all messages in such network must be encrypted to the same length in order to keep the attacker from distinguishing different messages by their content or length. The *anonymity set* for any particular message attacked by the eavesdropper is the set of all network participants that have sent messages within a certain time window before the attacked message was received. This time window of course depends on latency characteristics and node configurations of the network itself.

Sender anonymity against computationally restricted eavesdroppers can be achieved by ►MIX networks [1]. Against computationally unrestricted eavesdroppers it can be achieved by ►DC networks [2, 3].

Note that sender anonymity is weaker than sender unobservability, where the attacker cannot even determine whether or not a participant sends a message. Sender unobservability can be achieved with ►MIX networks and ►DC networks by adding dummy traffic.

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
2. Chaum D (1985) Security without identification. Transaction systems to make big brother obsolete. Commun ACM 28(10): 1030–1044
3. Chaum D (1988) The dining cryptographers problem. Unconditional sender and recipient untraceability. J Cryptol 1(1):65–75

Sensor Code Attestation

Yi Yang, Sencun Zhu
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Synonyms

Program integrity verification; Software-based attestation

Related Concepts

►Remote Attestation; ►Sensors

Definition

Sensor code attestation is a way to verify the integrity of the code running on sensors through contactless software-based attestation.

Background

Sensors that operate in an unattended or hostile environment often suffer from break-in compromises, because their low costs do not allow the use of expensive tamper-resistant hardware. Besides the exposure of secret information (e.g., cryptographic keys), compromised sensors will often be reprogrammed with malicious code to launch all kinds of insider attacks.

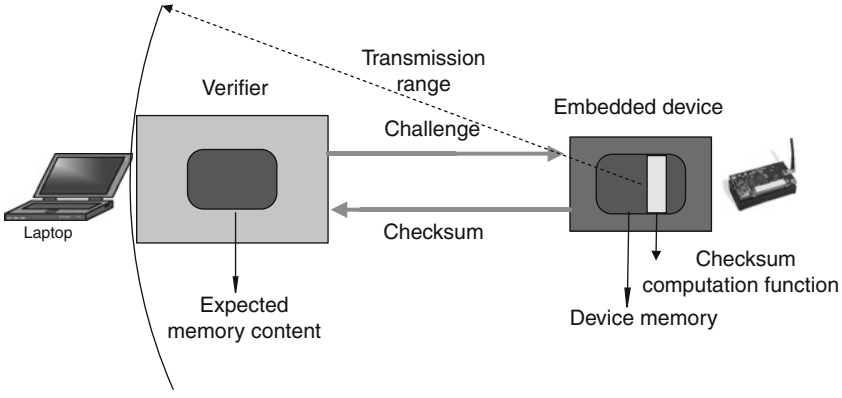
Very desirably, if corrupted nodes could be identified and further revoked in a timely manner, the potential damages caused by them could be minimized. Intrusion detection mechanisms such as watchdog and reputation-based system have been proposed to detect abnormal nodes. However, these behavior-based detections are error-prone, because they rely on accurate observation and reasoning of node's misbehavior.

Theory

Several sensor code attestation techniques have been proposed to verify the integrity of the code running on an embedded device (e.g., sensor) without physical access to the device or assistance of secure hardware. These techniques use a challenge–response protocol between a trusted external verifier and an interrogated device.

In SWATT [1], a verifier in the transmission range of the device sends a randomly generated number as the challenge. Upon receiving this challenge, the interrogated device traverses its memory in a pseudorandom fashion while recursively computing a cryptographic checksum over each traversed memory space, and responds to the verifier with the final checksum (Fig. 1). The verifier can validate the result because it knows the expected memory image, so that it can locally precompute the correct answer and estimate an upper bound for the response time. The pseudorandom memory traversal algorithm is designed in such a way that a tampered device either responds with a wrong answer or takes distinguishable longer time than a legitimate device does, leaving itself being detected.

Also, remote software attestation [2] was proposed for wireless sensors by sending a piece of attestation code from the base station to a remote sensor node. A potential attack here is that a compromised sensor may modify the attestation code. To address this attack, the attestation code is



Sensor Code Attestation. Fig. 1 Software-based attestation

obfuscated to make static analysis of the code difficult for an adversary and also randomized to make it hard for the adversary to predict the routine or precompute the results. Different from SWATT, this approach is not dependent on the response time difference to distinguish between a benign node and a compromised node.

SWATT guarantees detection of malicious programs probabilistically, thus requiring a large number of memory accesses to achieve longer time difference and a high detection rate. To reduce the computation overhead, program integrity verification (PIV) [3] constructs lightweight, randomized hash computation to verify the integrity of the program. In PIV, a new hash algorithm is generated for each attestation, and verification servers are widely distributed to verify hashes received from attested nodes.

To reprogram a sensor without being caught in an attestation, the attacker needs to keep a copy of the original code somewhere in its memory space; also, for a compromised node, its neighbors are the first and direct observers of its suspicious behaviors. Accordingly, in [4], first noises (pseudorandom numbers) are filled into the empty program memory of each node before deployment. Then, either the seed for generating the noise is distributed to multiple neighbors based on threshold secret sharing, or a random digest of the program memory content is assigned to each neighbor. When an attestation is triggered, multiple neighbors of a suspicious node collaborate in a challenge–response process and make a decision regarding the trustworthiness of this node in a distributed manner.

Open Problems

Various assumptions have been made in the existing schemes, such as the hardware (e.g., CPU and memory) of the device has not been tampered/enhanced or the code cannot be largely compressed. It is unclear how reasonably these assumptions hold in reality. Recently, it has been shown that various attacks exist [5]. How to solve them is an open problem in sensor code attestation.

Recommended Reading

1. Seshadri A, Perrig A, van Doorn L, Khosla P (2004) SWATT: software-based attestation for embedded devices. In: IEEE Symposium on Security and Privacy, Berkeley, California
2. Shaneck M, Mahadevan K, Kher V, Kim Y (2005) Remote software-based attestation for wireless sensors. In Engineering Semantic Agent Systems (ESAS), LNCS, vol 3813. Springer, Heidelberg, July 2005, pp 27–41
3. Park T, Shin KG (2005) Soft tamper-proofing via program integrity verification in wireless sensor networks. IEEE Trans Mobile Comput 4(3):297–309
4. Yang Y, Wang X, Zhu S, Cao G (2007) Distributed software-based attestation for node compromise detection in sensor networks. IEEE Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, China, October 2007
5. Castelluccia C, Francillon A, Perito D, Soriente C (2009) On the difficulty of software-based attestation of embedded devices. Proceedings of the 16th ACM conference on Computer and Communications Security (CCS), Chicago, Illinois, 2009

Sensor Key Establishment and Maintenance

►Sensor Key Management

Sensor Key Management

Yi Yang, Sencun Zhu
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Synonyms

Sensor key establishment and maintenance

Related Concepts

►Key Management; ►Sensor Networks

Definition

Key management involves all the operations related to cryptographic keys, including key generation, distribution, storage, update, etc.

Background

Sensor networks are powerful and economical solutions for both civilian and military applications, such as traffic monitoring and site surveillance, because their capability and flexibility are built on the top of low-cost and small-size sensors. Meanwhile, sensor networks are also vulnerable to various security attacks due to their extremely scarce resources and their unattended, hostile operating environments.

Cryptographic keys are used to encrypt/decrypt sensor data or to generate/verify authentication tags, in order to provide data confidentiality, integrity, and authentication. Therefore, key management is a fundamental security building block for sensor network security.

Theory

Providing security is particularly challenging in sensor networks due to the resource limitations of sensor nodes. As a specific example, it is not practical to use asymmetric cryptography in a sensor network where each node consists of a 7.8 MHz underpowered processor with only 4 KB of RAM space (Mica2 Motes). Thus, key management protocols for sensor networks are mostly based upon symmetric key algorithms.

An important design consideration for security protocols based on symmetric keys is the degree of key sharing between the nodes in the system. At one extreme, network-wide keys are used for encrypting data and for authentication. This key-sharing approach has the lowest storage costs and is very energy-efficient since no communication is required between nodes for establishing additional keys. However, it has the obvious security disadvantage that the compromise of a single node will reveal the global key.

At the other extreme, another key-sharing approach can be employed, in which all secure communication is based on keys that are shared pairwise between two nodes. From the security point of view, this approach is ideal since the compromise of a node does not reveal any keys that are used by the other nodes in the network. However, under this approach, each node will need a unique key for every other node that it communicates with. Moreover, in many sensor networks, the immediate neighbors of a sensor node cannot be predicted in advance; consequently, these pairwise shared keys will need to be established after the network is deployed.

In practice, different types of messages exchanged between sensor nodes have different security requirements, and that a single keying mechanism is not suitable for meeting these different security requirements. Therefore, typically four types of keys are needed in sensor networks [1].

Individual Key: Every node has a unique key that it shares with the base station. This key is used for secure communication between the node and the base station. For example, a node can use its individual key to compute message authentication codes (MACs) for its sensed readings if the readings are to be verified by the base station. A node may also send an alert to the base station if it observes any abnormal or unexpected behavior of a neighboring node. Similarly, the base station can use this key to encrypt any sensitive information, e.g., keying material or special instruction that it sends to an individual node.

Group Key: This is a globally shared key that is used by the base station for encrypting messages that are broadcast to the whole group. For example, the base station issues missions and sends queries and interests. Note that from the confidentiality point of view there is no advantage to separately encrypting a broadcast message using the individual key of each node. However, since the group key is shared among all the nodes in the network, an efficient rekeying mechanism is necessary for updating this key after a compromised node is revoked.

Cluster Key: A cluster key is a key shared by a node and all its neighbors, and it is mainly used for securing locally broadcast messages, e.g., routing control information or securing sensor messages, which can benefit from passive participation. Researchers have shown that in-network processing techniques, including data aggregation and passive participation, are very important for saving energy consumption in sensor networks. For example, a node that overhears a neighboring sensor node transmitting the same reading as its own current reading can elect to not transmit the same. In responding to aggregation operations such as MAX, a node can also suppress its own reading if its reading is not larger than an overheard one. Clearly, for passive participation to be feasible, sensor nodes should be able to decrypt or verify some classes of messages, e.g., sensor readings, transmitted by their neighbors. This requires that such messages be encrypted or authenticated by a locally shared key. As such, each node is provided with a unique cluster key shared with all its neighbors for securing its messages. Its neighbors use the same key for decrypting or verifying its messages.

Pairwise Key: Every node shares a pairwise key with each of its immediate neighbors. Pairwise keys are used for securing communications that require privacy or source authentication. For example, a node can use its pairwise keys to secure the distribution of its cluster key to its neighbors, or to secure the transmission of its sensor readings to an aggregation node. Note that the use of pairwise keys precludes passive participation.

Among all these keys, since individual and group keys could be preloaded into every sensor and cluster key could be established through pairwise key, pairwise key management is the focus of following discussion. In general, there are two ways to establish pairwise keys: *deterministic* and *probabilistic*.

In a deterministic scheme, such as the Blundo scheme [2], the pairwise key is established in a deterministic way. To be more specific, the key server first randomly generates a symmetric bivariate k-degree polynomial $f(x,y) = \sum_{i,j=0}^{k} a_{ij}x^i y^j$ over a finite field Fq, where q is a prime number that is large enough to accommodate a cryptographic key. A polynomial f(x, y) is said to be symmetric if f(x, y) = f(y, x). The key server computes f(i, y) for node i, and then loads node i with all the $k + 1$ coefficients (as a function of y). When two nodes i and j want to establish a pairwise key, they compute f(i, j) (or f(j, i), which is the same) by evaluating f(i, y) at point j and f(j, y) at point

i, respectively; f(i, j) serves as their pairwise key. The above scheme has been proved to be unconditionally secure and k-collusion-resistant.

A probabilistic scheme [3] typically has three phases: first, key setup prior to deployment; second, shared key discovery after deployment; third, path key establishment if two sensors do not share a common key in phase two. Specifically, prior to deployment, a ring of keys is distributed to each sensor node, each key ring consisting of randomly chosen keys from a large pool of keys that are generated off-line. Then, after deployment, two nodes can communicate to discover the shared keys between them. Because of the probabilistic nature of this scheme, shared keys may not exist between some pairs of nodes. In this case, some other nodes in the path can help them to establish a key. There are also improvement schemes, e.g., by leveraging deployment knowledge in pairwise key establishment, to enhance the probability of key sharing [4].

Open Problems

Symmetric-key-based schemes are widely used as they have relatively low computational complexity, which are suitable for wireless sensor networks that are very limited in resources. Public key schemes can provide simpler solutions with much stronger security guarantee, but it was commonly believed that public key schemes are too computationally expensive for wireless sensor networks. However, recent research [5–7] has shown that public key schemes (e.g., based on elliptic curve cryptography (ECC) [8]) are no longer impractical for sensor networks, although their applications are still limited. How to deploy and implement public key cryptography for general applications in wireless sensor networks will be an open problem in the near future.

Recommended Reading

1. Zhu S, Setia S, Jajodia S (2003) LEAP: efficient security mechanisms for large-scale distributed sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03), Washington DC, October 2003
2. Blundo C, Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M (1993) Perfectly-secure key distribution for dynamic conferences. In: Advances in Cryptology CRYPTO 92, LNCS 740, pp 471–486
3. Eschenauer L, Gligor VD (2002) A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, 2002
4. Du W, Deng J, Han YS, Chen S, Varshney PK (2004) A key management scheme for wireless sensor networks using deployment knowledge. In: Proceedings of the IEEE INFOCOM 2004, the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7–11, 2004
5. Wander A, Gura N, Eberle H, Gupta V, Shantz S (2005) Energy analysis of public-key cryptography on small wireless devices. In IEEE Pervasive Computing and Communication (PerCom), Kauai Island, Hawaii, March 2005
6. Du W, Wang R, Ning P (2005) An efficient scheme for authenticating public keys in sensor networks. In: Proceeding of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Urbana-Champaign, Illinois, May 2005, pp 58–67
7. Zhang J, Varadharajan V (2009) Wireless sensor network key management survey and taxonomy. J Network Comput Appl
8. Liu A, Ning P (2008) TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In: Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, St. Louis, Missouri, pp 245–256, April 2008

SEPA

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Single euro payments area

Definition

The Single Euro Payments Area (SEPA) will be the area where citizens, companies, and other economic participants make and receive payments in euro, whether between or within national boundaries, under the same basic conditions, rights, and obligations. In the long term, the uniform SEPA payment instruments are expected to replace national euro payment systems now being operated in Europe.

Background

The European Payments Council (EPC) is the decision-making and coordination body of the European banking industry in relation to payments. The EPC develops the payment schemes and frameworks necessary to realize the *Single Euro Payments Area* (SEPA). SEPA is an EU integration initiative in the area of payments designed to achieve the completion of the EU internal market and monetary union.

SEPA is an EU-wide policy-maker-driven integration initiative in the area of payments designed to achieve the completion of the EU internal market and monetary union. Following the introduction of euro notes and coins in 2002, the political drivers of the SEPA initiative – EU governments, the European Commission and the European Central Bank – focused on harmonizing the euro payments market. Integrating the multitude of national payment systems existing today is a natural step toward

making the euro a truly single and fully functioning currency. SEPA will become a reality when a critical mass of euro payments has migrated from legacy payment instruments to the new SEPA payment instruments.

Applications

The European banking industry has defined SEPA schemes for credit transfers and direct debits together with a SEPA data format based on global ISO standards. The SEPA Credit Transfer scheme was successfully launched in January 2008. The SEPA Core Direct Debit scheme and the SEPA Business to Business Direct Debit scheme went live in November 2009, the point in time when EU member states have adopted a common legal framework for payments. Both Debit schemes include the option to use electronic mandates (e-mandates) as well as or instead of the customary paper mandates. As of November 2009, banks gradually roll out SEPA Direct Debit services. In a step-by-step process all banks in the euro area offering direct debit services today will become reachable for SEPA Core Direct Debit by November 2010. The recent EU Regulation on cross-border payments in euro mandates the timelines for banks to create reachability for European direct debits.

For payment cards, a SEPA Cards Framework has been agreed and is in the process of being implemented by banks, card schemes, and card processors. It describes a general purpose card for euro payments including principles for banks, card schemes, card service providers, and other stakeholders.

Recommended Reading

1. http://ec.europa.eu/internal_market/payments/sepa/index_en.htm
2. http://www.europeanpaymentscouncil.eu/
3. http://www.ecb.int/paym/sepa/html/links.en.html

Separation of Duties

Mary Ellen Zurko[1], Richard T. Simon[2]
[1]IBM Software Group Lotus Live Security Architecture and Strategy, Westford, MA, USA
[2]Harvard Medical School, Center for Biomedical Informatics ,Cambridge, MA, USA

Synonyms

Dynamic separation of duties; History-based separation of duties; Operational separation of duties; Segregation of duties; Static separation of duties; Strong exclusion; Weak exclusion

Related Concepts

►Least Privilege; ►RBAC; ►Roles

Definition

Separation of Duty is a security principle used to formulate multi-person control policies, requiring that two or more different people be responsible for the completion of a task or set of related tasks. The purpose of this principle is to discourage fraud by spreading the responsibility and authority for an action or task over multiple people, thereby raising the risk involved in committing a fraudulent act by requiring the involvement of more than one individual. A frequently used example is the process of creating and approving purchase orders. If a single person creates and approves purchase orders, it is easy and tempting for them to create and approve a phony order and pocket the money; if different people must create and approve orders, then committing fraud requires a conspiracy of at least two, which raises the risk of disclosure and capture significantly.

Background

Separation of Duty is a fundamental principle in computer security. In 1975, Saltzer and Schroeder [8] defined "separation of privilege" as one of the eight design principles for the protection of information in computer systems. They credit R. Needham with making the following observation in 1973: a protection mechanism that requires two keys to unlock is more robust and flexible than one that requires only a single key. No single accident, deception, or breach of trust is sufficient to compromise the system.

Clark and Wilson's [2] commercial security policy for integrity identified Separation of Duty as one of the two major mechanisms to counter fraud and error while ensuring the correspondence between data objects within a system and the real world objects they represent. At the policy level, processes were divided into steps, with each step being performed by a different person. Thus Separation of Duty is tightly tied to application semantics or commands. Clark and Wilson suggested further safeguarding against collusion by random selection of the sets of people to perform some operation, so that any proposed collusion is only safe by chance.

At the formal model level, Clark and Wilson tasked the security administrator with the job of maintaining Separation of Duty requirements while granting users the ability to run Transformation Procedures against Constrained Data Items. Since this difficult task was done by hand, it was rarely repeated and encoded a static representation of who may perform which actions. Moreover, the model

specified that an agent who can certify the use of a Transformation Procedure (TP) may not execute that TP – but the identity of the certifier was not part of the model, so this was another Separation of Duty constraint that the security administrator had to remember and enforce.

Baldwin's [1] Named Protection Domains (NPDs) implemented many of the concepts used in role-based systems today. A NPD was a named, hierarchical grouping of database privileges and users. To help enforce Separation of Duty, a user could have only one of these NPDs activated at any time. The security administrator determined which NPDs could be activated, but there were no further restrictions on the graph of NPDs (other than it be acyclic). Thus, one activatable NPD could contain multiple activatable and nonactivatable NPDs. While the activation restriction meant that a user could be in only one role (NPD) at a time, the security administrator could set up arbitrarily complex roles.

Sandhu's work on Transaction Control Expressions introduced notation for Dynamic Separation of Duty. Roles were used to specify who can issue which transaction steps (much like NPDs) [10]. However, in Sandhu's model each user executing a step in a transaction had to be different. To enforce this, the history of the execution of each transaction was maintained. The constraints specifying the roles that could execute each step were associated with an object. These constraints turned into the history specifying which user executed each step on that object. Hierarchical roles were specified either on an object or via a global notation. A weighted voting syntax allowed the specification of multiple person authorizations on a particular step on a particular object.

Nash and Poland's [7] study of a portable security device used in the commercial world raised a number of new issues around Separation of Duty. The system defined two disjoint groups of authorizing officers, and each day one or two officers from each of those groups were chosen as officers of the day. This was the first example of the utility of specifying cardinality for a particular role and of time-based roles. Each transaction had to be authorized by one of those officers from each of the two groups. While the device enforced Separation of Duty between the manager who chooses officers and the officers themselves, the documentation suggested these roles could be shared in small companies. In their general discussion of Separation of Duty, Nash and Poland stated that neither TCSEC [3] mechanisms nor Clark and Wilson's original proposal allowed implementation of common commercial schemes for Separation of Duty. Nash and Poland proposed the notion of "object-based Separation of Duty," which forced every transaction against an object to be by a different user. They suggested using Sandhu's Transaction Control Expressions to maintain the history of an object's transactions.

Work by Sandhu and others [9, 11, 12] on defining role-based access controls (RBAC) with more precision acknowledged that there might be a need for users to hold multiple roles (that are not connected hierarchically) at the same time. They recognized a need for a way to place limits on how more powerful roles are combined with less powerful ones. They also suggested mechanisms such as time constraints on roles to limit the amount of damage that might be done with misuse or intrusion.

Ferraiolo, Cugini, and Kuhn's [4] paper on RBAC presented the beginnings of a formal model of RBAC. They defined three kinds of Separation of Duty. The first two were Static Separation of Duty and Dynamic Separation of Duty. These variants were presented in previous work. The third kind was Operational Separation of Duty which introduced the notion of a "business function" and the set of operations required for that function; a business function resembles the notions of task and task unit in Thomas and Sandhu [14]. The formal definition of Operational Separation of Duty stated that no role can contain the permissions to execute all of the operations necessary to a single business function. This forces all business functions to require at least two roles to be used for their completion. The informal description of Operational Separation of Duty assumes the roles involved have disjoint memberships (Static Separation of Duty), so that no single person has access to all the operations in a business function.

Although Separation of Duty is easy to understand, it is hard to implement in ways that reflect the actual operation of human organizations. Zurko, Simon, and Sanfillipo [16, 17] identified the need to incorporate the principle of user-centered security into Separation of Duty.

The Spatial RBAC model [5] addresses separation of duty as it applies to policies that make roles and permissions available based on the user's location. SRBAC allows users to be authorized to mutually exclusive roles if they cannot be executed in the same location.

Adding another layer of policy on top of separation of duty, a workflow security model, W1-RBAC [15], allows for controlled overriding of those constraints. The model allows for different levels of importance on constraints. This produces a consistent model at a specific level of compliance, identified by the highest importance level of a constraint that can be overridden. It also introduces the notion of an instance of a process so that separation of duty can apply to a specific workflow instance.

Simon and Zurko [13] expanded the scope of Separation of Duty by introducing History-Based Separation

of Duty, which allowed policies closer to those found in human organizations.

Gligor, Gavrila, and Ferraiolo [6] provided formal definitions of all the various forms of Separation of Duty.

Applications

The two broadest categories of Separation of Duty variations are *strong exclusion* or *Static Separation of Duty* and *weak exclusion* or *Dynamic Separation of Duty*. Strong exclusion represents a single variation whereas the more realistic and useful weak exclusion contains several variations. These variations are expressed as constraints on different aspects of RBAC:

- Constraints on role membership: overlap in membership is constrained.
- Constraints on role activation: legitimate members of the role may be prevented from assuming the role, because of constraints on when the role may be used.
- Constraints on role use: users who have assumed the role may be restricted in how it is used.

Strong Exclusion (Static Separation of Duty)

Strong exclusion is also called Static Separation of Duty and is the simplest variation of Separation of Duty. Two roles are strongly exclusive if no one person is ever allowed to perform in both of these roles. In other words, the two roles have no shared principals. Strong exclusion may be implemented using only controls over the membership of roles.

Though it has the advantage of simplicity, Static Separation of Duty is not a practical or realistic variation of Separation of Duty, because it does not reflect the actual functioning of human organizations. Users often have legitimate reasons for wanting or needing to act in two strongly exclusive roles, and careful construction of security policy can ensure that these "violations" are secure.

Because Static Separation of Duty is too rigid to be a satisfactory control, a number of other variations have been defined that more closely mimic the functioning of human organizations.

Weak Exclusion (Dynamic Separation of Duty)

Weak exclusion, or Dynamic Separation of Duty, provides a richer set of possible policies by controlling the activation and use of roles. Weak exclusion allows users to act in roles that would be strongly exclusive in static systems, as long as constraints are satisfied that eliminate or reduce the possibility of fraud. Weak exclusion has several variations.

In describing the variations of weak exclusion the term *restricted roles* is used to refer to roles that have constraints on their membership, activation, or use.

Simple Dynamic Separation of Duty. In the simplest variation of Dynamic Separation of Duty, restricted roles may have common members, but users may not assume both roles at the same time. This variation by itself is sometimes called Dynamic Separation of Duty.

Object-Based Separation of Duty. Restricted roles may have common members, and those members may assume both roles at the same time, but no user may act upon a target that the user has previously acted upon.

Operational Separation of Duty. Restricted roles may have common members as long as the union of all the groups of actions in the roles does not contain all the actions in a complete business. This prevents any one person from performing all of the actions in the business task.

History-Based Separation of Duty. Two or more roles may have common members and the union of the actions granted by those roles may span the actions in a business task, but no role member is allowed to perform all the actions in a business task on the same target or collection of targets. This definition combines features of both Object-based Separation of Duty and Operational Separation of Duty to permit the flexibility to express the kinds of policies that may be used in human organizations.

History-Based Separation of Duty may be refined further by noting that some policies define sequences of allowed or required steps and that each step may consist of order-dependent or order-independent actions.

Order-Dependent. In some situations, the fraud preventive nature of Separation of Duty controls is not implemented simply by splitting duties between roles; the roles also must perform their actions in a particular order. For example, a purchase should be approved only if it has been *previously* created properly.

Order-Independent. Sometimes order is not important. For example, suppose the policy states that two different approvals are required to make a purchase transaction valid. The order of the approvals is not important (so long as both happen after the order has been created), but they must both happen.

Open Problems

The most pressing open problem in separation of duty is driving its acceptance and use in deployed applications and systems. Clark and Wilson [2] and Baldwin [3] motivated the need for separation of duty in real world use cases, however aligning even simple static separation of duty with deployed permission and role models is difficult.

Existing permissions and roles must be mapped to specific user tasks where separation of duty makes sense, and the restrictions on permission assignment and task execution must be made clear to administrators and users (see next section "Experimental Results").

Experimental Results

User testing in Zurko, Simon, and Sanfillipo [12] indicated that administrators are often unfamiliar with the concept of separation of duty. Administrators were asked to assign permissions while faced with a static separation of duty constraint. Some of the administrators considered working around that constraint to complete their task and assign the desired permissions. These early user experiments highlight some of the usability challenges that impact the acceptance of the use of separation of duty.

Recommended Reading

1. Baldwin RW (1990) Naming and grouping privileges to simplify security management in large databases. In: Proceedings of the 1990 IEEE symposium on security and privacy, Oakland, May 1990, pp 116–132
2. Clark DD, Wilson DR (1987) A comparison of commercial and military computer security policies. In: Proceedings of the 1987 IEEE symposium on security and privacy, Oakland, April 1987, pp 184–194
3. Department of Defense National Computer Security Center (1985) Department of Defense Trusted Computer Systems Evaluation Criteria. DoD 5200.28-STD
4. Ferraiolo D, Cugini J, Kuhn DR (1995) Role-based access control (RBAC): features and motivations. In: Proceedings of the 1995 computer security applications conference, New Orleans, Dec 1995, pp 241–248
5. Frode H, Oleshchuk V (2003) SRBAC: a spatial role-based access control model for mobile systems. Nordsec 2003, Gjøvik, Norway, 15–17 Oct 2003, pp 129–141
6. Gligor V, Gavrila S, Ferraiolo D (1998) On the formal definition of separation-of-duty policies and their composition. In: Proceedings of the 1998 IEEE symposium on security and privacy, Rockport, May 1998, pp 172–183
7. Nash MJ, Poland KR (1990) Some conundrums concerning separation of duty. In: Proceedings of the 1990 IEEE symposium on security and privacy, Oakland, May 1990, pp 201–207
8. Saltzer JH, Schroeder MD (1975) The protection of information in computer systems. Proc IEEE 63(9):1278–1308
9. Sandhu R (1990) Separation of duties in computerized information systems. In: Proceedings of the IFIP WG11.3 workshop on database security, Halifax, UK, Sept 1990
10. Sandhu R (1998) Transaction control expressions for separation of duties. In: Proceedings of the fourth aerospace computer security conference, Orlando, Dec 1988, pp 282–286
11. Sandhu R, Coyne E, Feinstein H, Youman C (1994) Role-based access control: a multi-dimensional view. In: Proceedings of the 10th annual computer security applications conference, Orlando, Dec 1994, pp 54–62
12. Sandhu R, Feinstein H (1994) A three tier architecture for role-based access control. In: Proceedings of the 17th NIST-NCSC national computer security conference, Baltimore, Oct 1994, pp 138–149
13. Simon RT, Zurko M (1997) Separation of duty in role-based environments. In: 10th IEEE computer security foundations workshop, June 1997
14. Thomas RK, Sandhu RS (1994) Conceptual foundations for a model of task-based authorizations. In: Proceedings of the computer security foundations workshop VII, Franconia, June 1994
15. Wainer J, Barthelmess P, Kumar A (2001) "W-RBAC – a workflow security model incorporating controlled overriding of constraints" Instituto de Computacao, Universidade Estual de Campinas, Technical Report IC-01-013, October 2001
16. Zurko M, Simon RT (1996) User centered security. In: Proceedings of the new security paradigms workshop, Lake Arrowhead, Sept 1996
17. Zurko M, Simon R, Sanfillipo T (1999) A user-centered, modular authorization service built on an RBAC foundation. In: Proceedings of the 1999 IEEE symposium on security and privacy, Oakland, May 1999, pp 57–71

Sequences

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Autocorrelation; ►Berlekamp–Massey Algorithm; ►Boolean Functions; ►Cross-Correlation; ►De Bruijn Sequences; ►Gap; ►Golomb's Randomness Postulates; ►Linear Complexity; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-Sequences); ►Run; ►Stream Cipher

Definition

Sequence is an ordered set of elements from an alphabet.

Theory

Sequences have many applications in modern communication systems, including signal synchronization, navigation, radar ranging, code-division multiple-access (CDMA) systems, random number generation, spread-spectrum communications, and cryptography, in particular in ►stream cipher systems.

In stream cipher systems it is essential to construct sequences with good random properties, long periods, and large ►linear complexity. To achieve many of these goals one often generates sequences using linear recurrence relations [1, 6, 7]. The period of a sequence $\{s_t\}$ is the smallest integer ε such that $s_{t+\varepsilon} = s_t$ for all t. The period of a generated sequence is completely determined by the characteristic polynomial of the sequence. For linear sequences, the

period of the sequences generated can easily be controlled, which make them good building blocks in stream cipher systems.

A linear recursion of degree n with binary coefficients is given by

$$\sum_{i=0}^{n} f_i\, s_{t+i} = 0,$$

where $f_i \in GF(2) = \{0,1\}$ for $0 < i < n$ and $f_0 = f_n = 1$. The *characteristic polynomial* of the recursion is defined by

$$f(x) = \sum_{i=0}^{n} f_i x^i.$$

The initial state $(s_0, s_1, \ldots, s_{n-1})$ and the given recursion uniquely determine the generated sequence. A linear shift register with a characteristic polynomial $f(x)$ of degree n generates 2^n different sequences corresponding to the 2^n different initial states and these form a vector space over $GF(2)$, which is denoted $\Omega(f)$.

The maximal period of a sequence generated by a linear shift register is at most $2^n - 1$. This follows since a sequence is completely determined by n-successive bits in the sequence and period 2^n is impossible since n successive zeros implies the all-zero sequence. Sequences with the maximal period $2^n - 1$ are called m-sequences. For example, with initial state $(s_0, s_1, s_2) = (001)$, then $f(x) = x^3 + x + 1$ generates the m-sequence 0010111.

Example 1 *Let the recursion be*

$$s_{t+4} + s_{t+3} + s_{t+2} + s_{t+1} + s_t = 0 \pmod 2$$

with characteristic polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$. *The sequences in* $\Omega(f)$ *consists of the* $2^4 = 16$ *sequences corresponding to the sequences* $\{(0), (00011), (00101), (01111)\}$ *and their cyclic shifts.*

To analyze properties of linear sequences, one associates a generating function $G(x)$ with the sequence $\{s_t\}$, and let

$$G(x) = \sum_{t=0}^{\infty} s_t x^t.$$

Let $f^*(x) = \sum_{i=0}^{n} f_{n-i} x^i$ be the *reciprocal polynomial* of the characteristic polynomial of $f(x)$ of the sequence. Then, compute the product

$$\begin{aligned} G(x)f^*(x) &= (s_0 + s_1 x + s_2 x^2 + \cdots) \\ &\quad \times (1 + f_{n-1}x + \cdots + f_1 x^{n-1} + x^n) \\ &= \sum_{t=0}^{\infty} c_t x^t. \end{aligned}$$

The coefficient c_{t+n} of x^{t+n} for any $t \geq 0$ becomes

$$c_{t+n} = \sum_{i=0}^{n} f_i s_{t+i} = 0$$

as a consequence of the recurrence relation. Hence,

$$G(x)f^*(x) = \phi^*(x)$$

for some polynomial $\phi^*(x)$ of degree at most $n - 1$. Its reciprocal polynomial $\phi(x)$ is given by

$$\begin{aligned} \phi(x) &= s_0 x^{n-1} + (s_1 + f_{n-1}s_0)x^{n-2} + \cdots \\ &\quad + (s_{n-1} + f_{n-1}s_{n-2} + \cdots + f_1 s_0) \\ &= \sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1-i} f_{i+j+1} s_j \right) x^i. \end{aligned}$$

There is a one-to-one correspondence between any sequence $\{s_t\}$ in $\Omega(f)$ and any polynomial $\phi^*(x)$ of degree $\leq n - 1$. All sequences generated by $f(x)$ can therefore be described by

$$\Omega(f) = \left\{ \frac{\phi^*(x)}{f^*(x)} \mid deg(\phi^*(x)) < deg(f) = n \right\}.$$

For example, the m-sequence 0010111 in $\Omega(x^3 + x + 1)$ can be written as

$$\begin{aligned} \frac{x^2}{1 + x^2 + x^3} &= x^2 + x^4 + x^5 + x^6 + x^9 + x^{11} + x^{12} + \cdots \\ &= (x^2 + x^4 + x^5 + x^6)(1 + x^7 + x^{14} + \cdots. \end{aligned}$$

In particular a simple consequence of the above description of $\Omega(f)$ is that $\Omega(f) \subset \Omega(g)$ if and only if $f(x)$ divides $g(x)$.

The generating function $G(x)$ for a periodic sequence of period ε can be written as

$$\begin{aligned} G(x) &= (s_0 + s_1 x + \cdots + s_{\varepsilon-1} x^{\varepsilon-1}) \\ &\quad \times (1 + x^{\varepsilon} + x^{2\varepsilon} + \cdots) \\ &= \frac{s_0 + s_1 x + \cdots + s_{\varepsilon-1} x^{\varepsilon}}{1 - x^{\varepsilon}}. \end{aligned}$$

Combining the two expressions for $G(x)$ leads to the identity

$$(x^{\varepsilon} - 1)\phi(x) = \sigma(x) f(x),$$

where $\sigma(x) = s_0 x^{\varepsilon-1} + s_1 x^{\varepsilon-2} + \cdots + s_{\varepsilon-1}$, which contains all the information of a period of the sequence.

The *period of the polynomial* $f(x)$ is the smallest positive integer e such that $f(x)$ divides $x^e - 1$. The importance of the period e of $f(x)$ is that in order to find the period of all the sequences in $\Omega(f)$, it is enough to find the period of $f(x)$.

Since $f(x)$ divides $x^e - 1$, it follows that $\Omega(f) \subset \Omega(x^e - 1)$, the set of sequences where $s_{t+e} = s_t$, i.e., of period dividing e. Hence, all the sequences generated by $f(x)$ has period dividing e. Let the sequence $\{s_t\}$ correspond to the polynomial $\phi(x)$. If $gcd(f(x), \phi(x)) = 1$, then as a consequence of the identity $(x^\varepsilon - 1)\phi(x) = \sigma(x)f(x)$, it follows that $\{s_t\}$ has *smallest* period e, since in this case $f(x)$ must divide $x^\varepsilon - 1$ and thus $\varepsilon \geq e$, which implies that $\varepsilon = e$.

In particular when $f(x)$ is an irreducible polynomial of period e, then all the nonzero sequences in $\Omega(f)$ have period e. For example, the polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$ in Example 1 is irreducible and divides $x^5 + 1$ and has period 5, and therefore all nonzero sequences in $\Omega(f)$ have period 5.

To determine the cycle structure of $\Omega(f)$ for an arbitrary polynomial $f(x)$ that can be factored as $f(x) = \prod f_i(x)^{k_i}$, $f_i(x)$ irreducible, one first need to determine the cycle structure of $\Omega\left(f_i^{k_i}\right)$ and then the cycle structure for $\Omega(gh)$ when $gcd(g, h) = 1$.

Cycle structure of $\Omega(f^r)$. Let $f(x)$ be an irreducible polynomial of period e. Let k be defined such that $2^k < r \leq 2^{k+1}$. Then $\Omega(f^r) \backslash \Omega(f)$ contains

$$2^{n2^j} - 2^{n2^{j-1}}$$

sequences of period $e2^j$ for $j = 1, 2, \ldots, k$ and

$$2^{nr} - 2^{n2^k}$$

sequences of period $e2^{k+1}$.

Example 2 *Let $f(x) = x^3 + x + 1$ be the characteristic polynomial with $e = 7$, that generates an m-sequence. The number of sequences of each period in $\Omega(f^3)$ is therefore*

Number	1	7	56	448
Period	1	7	14	28

Cycle structure of $\Omega(gh)$ when $gcd(g, h) = 1$. In this case, it can be shown that each sequence $\{s_t\}$ in $\Omega(gh)$ can be written uniquely

$$\{s_t\} = \{u_t\} + \{v_t\},$$

where $\{u_t\} \in \Omega(g)$ and $\{v_t\} \in \Omega(h)$. Further, the period of the sum $\{u_t\} + \{v_t\}$ is equal to the ▶least common multiple of the period of the two sequences, i.e.,

$$per(s_t) = lcm(per(u_t), per(v_t)).$$

To find the cycle structure of $\Omega(gh)$, suppose $\Omega(g)$ contains d_1 cycles of length λ_1 and $\Omega(h)$ contain d_2 cycles of length λ_2. Add in all possible ways the corresponding $d_1\lambda_1$ sequences from $\Omega(g)$ and the $d_2\lambda_2$ sequences from $\Omega(h)$. This gives $d_1\lambda_1 d_2\lambda_2$ distinct sequences all of period $lcm(\lambda_1, \lambda_2)$. Formally this can be written as $[d_1(\lambda_1)][d_2(\lambda_2)] = [d(\lambda)]$, where $d = d_1 d_2 gcd(\lambda_1, \lambda_2)$ and $\lambda = lcm(\lambda_1, \lambda_2)$.

Example 3 *Let $f_1(x) = x^3 + x + 1$ and $f_2(x) = x^4 + x^3 + x^2 + x + 1$. The cycle structure of $\Omega(f_1)$ can be written $[1(1) + 1(7)]$, and similarly for $\Omega(f_1)$ as $[1(1) + 3(5)]$. Combining the cycle structure as described above, gives the cycle structure $[1(1) + 1(7) + 3(5) + 3(35)]$.*

The discussion above shows that the period of all sequences in $\Omega(f)$ is completely determined from the periods of the divisors of $f(x)$. This way of controlling the periods is one of the main reasons for using linear recursions as building blocks in stream ciphers.

The sequence $\{s_t\}$ can be expressed in terms of the zeros of its characteristic polynomial $f(x)$ of degree n. In the case when the zeros of $f(x)$ are simple, which is the case when the sequence has odd period, then $\{s_t\}$ has a unique expansion in the form

$$s_t = \sum_{i=1}^{n} a_i \alpha_i^t$$

for some constants a_i and where α_i, $1 \leq i \leq n$ are the zeros of $f(x)$.

The main problem with linear recursions in cryptography is that it is easy to reconstruct a sequence $\{s_t\}$ generated by a characteristic polynomial $f(x)$ of degree n from the knowledge of $2n$ consecutive bits in $\{s_t\}$, since this gives a system of n equations for determining the unknown coefficients of $f(x)$. The ▶Berlekamp–Massey algorithm is an efficient method for finding $f(x)$ in this way. Several methods exist to increase the linear span, i.e., the smallest degree of the linear recursion that generates the sequence. Further mentioned are a few simple constructions obtained by multiplying sequences.

Let $\{u_t\}$ and $\{v_t\}$ be two sequences of odd period. Then, $u_t = \sum a_i \alpha_i^t$, where α_i, $1 \leq i \leq n$, are the zeros of the characteristic polynomial of $\{u_t\}$ and $v_t = \sum b_j \beta_j^t$, where β_j, $1 \leq j \leq m$ are the zeros of the characteristic polynomial of $\{v_t\}$. Then the sequence $\{w_t\} = \{u_t v_t\}$ can be written as

$$w_t = \sum a_i b_j (\alpha_i \beta_j)^t.$$

This shows that $\{w_t\}$ is generated by the polynomial with, at most, the nm different zeros $\alpha_i\beta_j$ for $1 \leq i \leq n$, $1 \leq j \leq n$. If $gcd(per(u_t), per(v_t)) = 1$, then it can be shown that $per(w_t) = per(u_t)per(v_t)$. However, the sequence $\{w_t\}$ will in general not be *balanced* even if this is the case for $\{u_t\}$ and $\{v_t\}$. This follows since $w_t = 1$ if and only if $u_t = v_t = 1$, and thus only 1/4 of the elements in $\{w_t\}$ will be 1s when $\{u_t\}$ and $\{v_t\}$ are balanced.

Often one considers sequences of the form

$$w_t = s_{t+\tau_1} s_{t+\tau_2} \ldots s_{t+\tau_k},$$

where s_t is an m-sequence. A closer study of the zeros of the characteristic polynomial of $\{w_t\}$ shows that the linear span is at most $\sum_{i=1}^{k} \binom{n}{i}$ and frequently the equality holds.

Every Boolean function in n variables, $f(x_1, x_2, \ldots, x_n)$, can be written uniquely as the sum

$$f(x_1, x_2, \ldots, x_n) = u_0 + \sum_{i=1}^{n} u_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} x_i x_j + \cdots + u_{12\ldots n} x_1 x_2 \cdots x_n$$

with binary coefficients (see the algebraic normal form in ►Boolean functions).

One can determine the linear span obtained by combining n in different m-sequences $\left\{a_t^{(1)}\right\}, \left\{a_t^{(2)}\right\}, \ldots, \left\{a_t^{(n)}\right\}$ with characteristic polynomials of pair-wise relative prime degrees $e_1, e_2, \ldots, e_n$ using a Boolean combining function. From the Boolean function $f(x_1, x_2, \ldots, x_n)$ one can easily construct the sequence $w_t = f\left(a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(n)}\right)$. Then the linear span of the combined sequence is equal to $f(e_1, e_2, \ldots, e_n)$, evaluated over the integers.

It is important in applications of sequences in communication systems as well as in stream cipher systems to generate sequences with good auto- and cross-correlation properties.

Let $\{u(t)\}$ and $\{v(t)\}$ be two binary sequences of period e. The ►cross correlation of the sequences $\{u(t)\}$ and $\{v(t)\}$ at shift τ is defined as

$$C_{u,v}(\tau) = \sum_{t=0}^{e-1} (-1)^{u_{t+\tau} - v_t},$$

where the sum $t + \tau$ is computed modulo e. In the case when the two sequences are the same, this is called the autocorrelation at shift τ.

For synchronization purposes one prefers sequences with low absolute values of the maximal out-of-phase ►autocorrelation, i.e., $|C_{u,u}(\tau)|$ should be small for all values of $\tau \neq 0 \pmod{e}$.

Applications

Let $\mathcal{F}$ be a family consisting of M sequences

$$\mathcal{F} = \{s_i(t) : i = 1, 2, \cdots, M\},$$

where each sequence $\{s_i(t)\}$ has period e.

The cross correlation between two sequences $\{s_i(t)\}$ and $\{s_j(t)\}$ at shift τ is denoted by $C_{i,j}(\tau)$. In CDMA applications it is desirable to have a family of sequences with certain properties. To facilitate synchronization, it is desirable that all the out-of-phase autocorrelation values $(i = j, \tau \neq 0)$ are small. To minimize the interference due to the other users in a multiple access situation, the cross-correlation values $(i \neq j)$ must also be kept small. For this reason the family of sequences should be designed to minimize

$$C_{\max} = \max\{|C_{i,j}| : 1 \leq i, j \leq M, \text{ and either } i \neq j \text{ or } \tau \neq 0\}.$$

For practical applications in communication systems one needs a family $\mathcal{F}$ of sequences of period e, such that the number of users $M = |\mathcal{F}|$ is large and simultaneously $C_{\max}$ is small [4, 5]. Also in stream ciphers [3] it is of importance that the generated sequences have good autocorrelation and cross-correlation properties [2].

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco, 1967. Revised ed., Aegean Park Press, Laguna Hills, 1982
2. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge
3. Helleseth T (2009) Linear and nonlinear sequences and applications to stream ciphers. In: Luengo I (ed) Recent trends in cryptography, vol 477 of Contemporary mathematics, American Mathematical Society, Providence, Rhode Island, pp 21–46
4. Helleseth T, Vijay Kumar P (1998) Sequences with low correlation. In: Pless VS, Huffman WC (eds) Handbook in coding theory, vol II. Elsevier, Amsterdam, pp 1765–1853
5. Helleseth T, Vijay Kumar P (2002) Pseudonoise sequences. In: Gibson JD (ed) The communications handbook, 2nd edn. CRC Press, London, pp 8–1–8–12
6. Selmer ER (1966) Linear recurrence relations over finite fields. Department of Mathematics, University of Bergen, Norway
7. Zierler N (1959) Linear recurring sequences. J Soc Ind Appl Math 7(1):31–48

SERPENT

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►AES Candidate; ►Block Ciphers

Serpent is a 128-bit ►block cipher designed by Anderson et al. and first published in 1998 [1]. Later that year the cipher was slightly modified [2] and proposed as a candidate for the *Advanced Encryption Standard* (►Rijndael/AES). In 1999, it was selected as one of the five finalists of the AES competition.

Serpent is a 32-round ►substitution-permutation (SP) network operating on 128-bit blocks. Each round consists of a key mixing operation, a layer of 32 copies of a 4×4-bit S-box, and (except in the last round) a linear transformation. The replicated S-box differs from round to round and is selected from a set of eight different S-boxes. The last (incomplete) round is followed by a final key mixing operation. An additional bit permutation before the first round and after the last key mixing layer is applied to all data entering and leaving the SP network. The 128-bit subkeys mixed with the data in each round are generated by linearly expanding a 128-bit, 192-bit, or 256-bit secret key, and passing the result through a layer of S-boxes.

The initial and final permutations, the S-boxes, and the linear transformation have all been designed in order to allow and optimized implementation in software using the "bitslice" technique [3]. The idea is to construct a complete description of the cipher using only logical bit-operations (as in hardware) and then execute 32 (or 64) operations in parallel on a 32-bit (or 64-bit) processor.

Serpent is considered to have a rather high security margin. The best attacks published so far break about one-third of the rounds. Kelsey, Kohno, and Schneier [7] presented a first attack breaking nine rounds with a time complexity slightly faster than ►exhaustive key search. This amplified ►boomerang attack was improved and extended by one round by Biham et al. [5]. The best attacks so far are the linear and the differential-linear attacks presented in [4] and [6]. Both break 11 rounds out of 32.

Recommended Reading

1. Anderson RJ, Biham E, Knudsen LR (1998) Serpent: a new block cipher proposal. In: Vaudenay S (ed) Fast software encryption (FSE'98), Paris, 23–25 March 1998. Lecture notes in computer science, vol 1372. Springer, Berlin, pp 222–238
2. Anderson RJ, Biham E, Knudsen LR (1998) Serpent: a proposal for the advanced encryption standard. In: Proceedings of the First AES Candidate Conference, 20–22 August 1998. National Institute of Standards and Technology, Gaithersburg
3. Biham E (1997) A fast new DES implementation in software. In: Biham E (ed) Fast software encryption (FSE'97), Haifa, 20–22 January 1997. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 260–272
4. Biham E, Dunkelman O, Keller N (2002) Linear cryptanalysis of reduced round Serpent. In: Matsui M (ed) Fast software encryption (FSE 2001), Yokohama, 2–4 April 2001. Lecture notes in computer science, vol 2355. Springer, Berlin, pp 16–27
5. Biham E, Dunkelman O, Keller N (2002) New results on boomerang and rectangle attacks. In: Daemen J, Rijmen V (eds) Fast software encryption (FSE 2002), Leuven, 4–6 February 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 1–16
6. Biham E, Dunkelman O, Keller N (2003) Differential-linear cryptanalysis of Serpent. In: Johansson T (ed) Fast software encryption (FSE 2003), Lund, 24–26 February 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 9–21
7. Kelsey J, Kohno T, Schneier B (2001) Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier B (ed) Fast software encryption (FSE 2000), New York, 10–12 April 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 75–93

Session Hijacking Attacks

Martin Johns
SAP Research CEC Karlsruhe, SAP AG, D-76131 Karlsruhe, Germany

Related Concepts

►Cross-site Scripting (XSS)

Definition

The term *Session hijacking attacks* refers to a class of attacks specific to Web applications. It describes situations in which the adversary impersonates a Web application's user through unauthorized usage of session credentials within adversary-controlled HTTP requests.

Background

The World Wide Web (WWW) as introduced by Tim Berners Lee in 1990 [1] is based on the communication protocol HTTP and the presentation language HTML. Originally, the WWW was proposed as a dedicated delivery mechanism for static hypertext documents. Consequently, HTTP defines a stateless request–response model that has no inherent session concept [2]. For this reason, the currently employed Web session tracking mechanisms are implemented within the Web applications. Hence, they are susceptible to application-level insecurities.

Theory

HTTP is a stateless protocol. Thus, HTTP has no protocol-level session concept. However, the introduction of dynamic Web application resulted in workflows that consist in a series of consecutive HTTP requests. Hence, the need for chaining HTTP requests from the same user into usage sessions arose. For this purpose, session identifiers (SID) were introduced. A SID is an alphanumerical value that is unique for the corresponding Web session. It is the Web application's duty to implement measures that include the SID in every HTTP request that belongs to the same Web session. A SID mechanism can be implemented by transporting the value either in form of an HTTP cookie or via HTTP parameters [2].

All requests that contain the SID are automatically regarded to belong to the same session and, thus, to the same user. After a user has authenticated himself against the application (e.g., by providing a valid password), his authorization state is also directly tied to his SID. Hence, the SID de facto becomes the user's authorization credential.

In general session hijacking attacks can be subclassified into two distinct sets [3]:

- Session hijacking via SID leakage: If the adversary is capable of obtaining the SID value, he can fully impersonate the victim by creating HTTP requests that contain the stolen SID. The Web application's server cannot distinguish these from genuine user-intended requests. Thus, this enables the attacker to completely impersonate the victim. SID leakage can be caused either by cross-site scripting (XSS) or through a man-in-the-middle, network-based attack.
- Session hijacking within the Web browser: In case of an XSS-based attack, the adversary is capable to execute arbitrary JavaScript in the victim's Web browser. Hence, he can conduct his actions completely within the victim's browser without the need for SID leakage. The actual hijacking attack is executed by injecting a JavaScript which in turn creates all HTTP requests that are necessary to satisfy the adversary intent. This technique is, for instance, used by fully automated, self-replicating XSS worms [4].

Open Problems

All currently utilized session tracking mechanisms are susceptible to session hijacking attacks in case of either an XSS vulnerability or a condition that causes the leakage of the session identifier. Current defensive approaches are either cumbersome to implement [3], concentrate exclusively on the subclass of SID leakage [5, 6], or solely rely on the prevention of XSS vulnerabilities (e.g., [7–9]). The creation of a secure session tracking mechanism that is native to HTTP and immune to browser-level hijacking attacks is still unsolved.

Recommended Reading

1. Berners-Lee T, Cailliau R (1990) WorldWideWeb: proposal for a HyperText Project, technical report, http://www.w3.org/Proposal
2. Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T (1999) Hypertext Transfer Protocol – HTTP/1.1, RFC 2616
3. Johns M (2006) SessionSafe: implementing XSS immune session handling. European Symposium on Research in Computer Security (ESORICS 2006), LNCS 4189. Springer, Berlin, pp 444–460
4. Kamkar S (2005) Technical explanation of the MySpace worm [online], http://namb.la/popular/tech.html
5. Kirda E, Kruegel C, Vigna G, Jovanovic N (2006) Noxes: a client-side solution for mitigating cross site scripting attacks. In Security Track of the 21st ACM Symposium on Applied Computing (SAC 2006), Dijon, France
6. Vogt P, Nentwich F, Jovanovic N, Kruegel C, Kirda E, Vigna G (2007) Cross site scripting prevention with dynamic data tainting and static analysis. In the 14th Annual Network and Distributed System Security Symposium (NDSS 2007), San Diego, California
7. Pietraszek T, Berghe CV (2005) Defending against injection attacks through context-sensitive string evaluation. Recent Advances in Intrusion Detection (RAID2005), Seattle, Washington
8. Livshits B, Lam MS (2005) Finding security vulnerabilities in Java applications using static analysis. Proceedings of the 14th USENIX Security Symposium, 2005, Baltimore, Maryland
9. Ter Louw M, Venkatakrishna VN (2009) Blueprint: robust prevention of cross-site scripting attacks for existing browsers. IEEE Symposium on Security and Privacy, May 2009, Oakland, Maryland

SHA

►SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)

SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)

Helena Handschuh
Computer Security and Industrial Cryptography Research Group, Katholieke Universiteit Leuven, Leuven - Heverlee, Belgium

Synonyms

Secure hash algorithm; SHA

Related Concepts

►Hash Functions; ►Message Digest; ►SHA-3 Competition

Definition

The SHA (Secure Hash Algorithm) Family currently designates a family of six different hash functions: SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [9, 10] published by the American National Institute of Standards

and Technology (NIST). They take variable length input messages and hash them to fixed-length outputs.

Background

Hash functions are functions which take a variable input message and compute a fixed-length message digest for each such message. This digest serves as a digital fingerprint allowing a receiver to check that the original message has not been altered during transmission. Hash functions are unkeyed cryptographic primitives which do not guarantee authenticity. They can be combined with a secret key to produce a message authentication code. They also serve as a cryptographic primitive used in digital signature schemes.

Theory

SHA-0, SHA-1, SHA-224, and SHA-256 operate on 512-bit message blocks at a time divided into 32-bit words, whereas SHA-384 and SHA-512 operate on 1024-bit blocks divided into 64-bit words. SHA-0 (the first published version of SHA since then replaced by SHA-1) and SHA-1 produce a message digest of 160 bits, SHA-224 of 224 bits, SHA-256 of 256 bits, SHA-384 of 384 bits, and SHA-512 of 512 bits, respectively. All six functions start by padding the message according to the so-called Merkle-Damgård strengthening technique. Next, the message is processed block by block by the underlying compression function. This function initializes an appropriate number of chaining variables to a fixed value to hash the first message block, and to the current hash value for the following message blocks. At each step i, the compression function updates in turn one of the chaining variables according to one message word W_i. As there are more steps in the compression function than words in a message block, an additional message schedule is applied to expand the message block. In the last step, the initial value of the chaining variable is added to each variable to form the current hash value (or the final one if no more message blocks are available). The following provides an overview of SHA-1, SHA-256, and SHA-512. SHA-0 is almost identical to SHA-1, SHA-224 to SHA-256, and SHA-384 to SHA-512.

Padding

The message is appended with a binary one and right-padded with a variable number of zeroes followed by the length of the original message coded over two binary words. The total padded message length must be a multiple of the message block size.

SHA-1 Compression Function

Five 32-bit chaining variables A, B, C, D, E are either initialized to

$$A \leftarrow IV_1 = 67452301_x$$
$$B \leftarrow IV_2 = \text{EFCDAB89}_x$$
$$C \leftarrow IV_3 = \text{98BADCFE}_x$$
$$D \leftarrow IV_4 = 10325476_x$$
$$E \leftarrow IV_5 = \text{C3D2E1F0}_x$$

for the first 512-bit message block or to the current hash value for the following message blocks. The first sixteen words of the message schedule are initialized to input message words. The following 64 message schedule words W_i are computed as

$$W_i \leftarrow (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1, 16 \le i \le 79$$

where "$\oplus$" represents bit-wise exclusive-or, and "X≪n" is the cyclic rotation of X to the left by n bits. Then the compression function works as follows:

for $i = 0$ to 79 do

$$T \leftarrow W_i + A \lll 5 + f_i(B, C, D) + E + K_i \mod 2^{32}$$
$$B \leftarrow A$$
$$C \leftarrow B \lll 30$$
$$D \leftarrow C$$
$$E \leftarrow D$$
$$A \leftarrow T$$

where the nonlinear functions f_i are defined by

$$f_{if}(X, Y, Z) = (X \wedge Y) \mid (\neg X \wedge Z), \quad 0 \le i \le 19$$
$$f_{xor}(X, Y, Z) = (X \oplus Y \oplus Z), \quad 20 \le i \le 39, \ 60 \le i \le 79$$
$$f_{maj}(X, Y, Z) = (X \wedge Y) \mid (X \wedge Z) \mid (Y \wedge Z), \quad 40 \le i \le 59$$

and the constants K_i by

$$K_i \leftarrow \text{5A827999}_x, 0 \le i \le 19$$
$$K_i \leftarrow 6ED9\text{EBA1}_x, 20 \le i \le 39$$
$$K_i \leftarrow \text{8F1BBCDC}_x, 40 \le i \le 59$$
$$K_i \leftarrow CA62C1D6_x, 60 \le i \le 79.$$

After 80 steps, the output value of each chaining variable is added to the previous intermediate hash value according to the feed-forward Davies–Meyer construction to give the

new intermediate hash value. When all consecutive message blocks have been hashed, the last intermediate hash value is the final overall hash function output.

SHA-0

The only difference between SHA-1 and SHA-0 is the fact that there is no left rotation by one bit in the message schedule of SHA-0. In other words, the 64 message schedule words W_i for SHA-0 are computed as follows:

$$W_i \leftarrow W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, \quad 16 \le i \le 79.$$

SHA-256 and SHA-512 Compression Functions

Eight chaining variables *A, B, C, D, E, F, G, H* are initialized to fixed values H_0 to H_7 for the first message block, and to the current intermediate hash value for the following blocks. The first sixteen w-bit words (where $w = 32$ for SHA-256 and $w = 64$ for SHA-512) of the message schedule are initialized to the input message words. The following $r - 16$ (where $r = 64$ for SHA-256 and $r = 80$ for SHA-512) message schedule words W_i are computed as

$$W_i = \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} \mod 2^{\omega}, \quad 16 \le i \le r-1$$

where σ_0 and σ_1 represent linear combinations of three rotated values of the input variable. Then the compression function works as follows:

for $i = 0$ to r do
$T_1 \leftarrow H + \sum_1(E) + f_{if}(E, F, G) + K_i + W_i \mod 2^{\omega}$
$T_2 \leftarrow \sum_0(A) + f_{maj}(A, B, C) \mod 2^{\omega}$
$H \leftarrow G$
$G \leftarrow F$
$F \leftarrow E$
$E \leftarrow D + T_1 \mod 2^{\omega}$
$D \leftarrow C$
$C \leftarrow B$
$B \leftarrow A$
$A \leftarrow T_1 + T_2 \mod 2^{\omega}$

where Σ_0 and Σ_1 again represent linear combinations of three rotated values of the input variable and K_i is a different w-bit constant for each step i. Finally, the output value of each chaining variable is added to the previous intermediate hash value according to the feed-forward Davies–Meyer construction to give the new intermediate hash value. When all consecutive message blocks have been hashed, the last intermediate hash value is the final overall hash function output.

SHA-224

The SHA-224 hash computations are exactly the same as those of SHA-256, up to the following two differences: The constants H_0 to H_7 used in SHA-224 are not the same as those used in SHA-256, and the SHA-224 output is obtained by truncating the final overall hash value to its 224 leftmost bits.

SHA-384

The SHA-384 hash computations are exactly the same as those of SHA-512, up to the following two differences: The constants H_0 to H_7 used in SHA-384 are not the same as those used in SHA-512, and the SHA-384 output is obtained by truncating the final overall hash value to its 6 leftmost words.

Applications

All six SHA functions belong to the MD4 type hash functions and were introduced by the American National Institute for Standards and Technology (NIST). SHA was published as a Federal Information Processing Standard (FIPS) in 1993. This early version is known as SHA-0. In 1994, a minor change to SHA-0 was made, and published as SHA-1 [9]. SHA-1 was subsequently standardized by ISO [6]. SHA-1 is extensively used in security applications and protocols such as TLS, SSL, PGP, SSH, S/MIME, and IPSEC. It was first designed to be incorporated into the Digital Signature Standard, also published by NIST. The following generation of SHA functions with much larger message digest sizes, namely, 256, 384, and 512 bits, was introduced in 2000 and adopted as a FIPS standard in 2002 [10] as well as an ISO standard in 2003 [6]. This generation of hash functions provides theoretical security levels against collision search attacks which are consistent with the security levels expected from the three standard key sizes of the Advanced Encryption Standard (►Rijndael/AES) (128, 192, and 256 bits). SHA-1 and SHA-256 also served as a basis for the design of the SHACAL block ciphers. The latest member of the family, namely SHA-224, was adopted in a Change Notice to FIPS 180-2 in 2004 and matches the expected security level of two-key triple-DES (112 bit keys).

Security Considerations

The first attack known on SHA-0 is by Chabaud and Joux [2]. They show that in about 2^{61} evaluations of the compression function, it is possible to find two messages hashing to the same value whereas a brute-force attack exploiting the birthday paradox requires about 2^{80} evaluations

in theory. In 2004, Biham and Chen introduce the neutral bit technique and find near-collisions on the compression function of SHA-0 [1] as well as collisions on reduced-round versions of SHA-1. In 2004, Joux, Carribault, Jalby, and Lemuet [7] first provide a full collision on SHA-0 using two four-block messages and requiring a complexity of 2^{51} compression function computations. In 2005, Wang, Yin, and Yu [13] announce full collisions on SHA-0 in 2^{39} hash operations and report that collisions on SHA-1 can be obtained in less than 2^{69} hash operations. In 2007, Joux and Peyrin [8] obtain collisions on SHA-0 in complexity 2^{32} hash operations using a technique derived from block cipher cryptanalysis, namely, amplified boomerang attacks. Saarinen [11] addresses the existence of slid pairs in SHA-1. Subsequent work by different research teams seems to suggest that collisions for SHA-1 can be found with as low as 2^{52} hash operations. De Canniere and Rechberger study collision resistance and preimage resistance for reduced-round versions of SHA-0 and SHA-1. In 2008, they show that preimages can be found for 49 out of 80 steps for SHA-0 and for 44 out of 80 steps for SHA-1 in time faster than exhaustive search [3]. In 2009, Aoki and Sasaki improve those numbers to 52 and 48 steps out of 80 respectively, using meet-in-the middle attacks. Both De Canniere-Rechberger and Joux-Peyrin currently find collisions in complexity respectively 2^{44} and 2^{39} for up to 70 out of 80 steps in SHA-1. No Collisions for the full SHA-1 hash function have been found as of the end of 2010. The first security analysis on SHA-256, SHA-384, and SHA-512 in 2003 is by Gilbert and Handschuh [4]. They show that collisions can be found with a reduced work factor for weakened variants of these functions. Subsequently, Hawkes and Rose show that second pre-image attacks are far easier than expected on SHA-256 [5]. The best collision search attack by Sanadhya and Sarkar published in 2008 is on 24 out of 64 steps [12]. The first colliding message pair for SHA-512 reduced to 24 steps is also by Sanadhya and Sarkar. Preimages for 42 step-reduced SHA-2 are presented by Aoki, Guo, Matusiewicz, Sasaki, and Wang in late 2009. However, none of these observations leads to actual attacks on SHA-256 and SHA-512 so far.

Open Problems

NIST opens a competition to develop a new cryptographic hash function called SHA-3 in late 2008. Out of the 64 received submissions, 51 meet the minimum submission requirements and are subsequently presented at the First SHA-3 Candidate Conference in Leuven, Belgium in early 2009. In summer 2009, NIST announces the selection of 14 Second Round Candidates. These are BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein. The list is further reduced to the five finalists BLAKE, Grøstl, JH, Keccak and Skein end 2010. The expected timeline for completing the competition process and for publishing the revised and augmented Hash Function Standard is by 2012.

Recommended Reading

1. Biham E, Chen R (2004) Near-collisions of SHA-0. In: Franklin M (ed) Advances in cryptology – CRYPTO 2004. Lecture notes in computer science, vol 3152. Springer, Berlin, pp 290–305
2. Chabaud F, Joux A (1998) Differential collisions in SHA-0. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 56–71
3. De Canniere C, Rechberger C (2008) Preimages for reduced SHA-0 and SHA-1. In: Wagner D (ed) Advances in cryptology – CRYPTO 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 179–202
4. Gilbert H, Handschuh H (2004) Security analysis of SHA-256 and sisters. In: Matsui M, Zuccherato R (eds) Selected areas in cryptography – SAC 2003. Lecture notes in computer science, vol 3006. Springer, Berlin, pp 175–193
5. Hawkes P, Rose G (2004) On Corrective Patterns for the SHA-2 Family. http://eprint.iacr.org/2004/207
6. ISO/IEC,10118-3 (2003) Information, technology – security techniques – hash-functions – Part 3: Dedicated hash-functions
7. Joux A, Carribault P, Jalby W, Lemuet C (2004) Collisions in SHA-0. Presented at the rump session of *CRYPTO 2004*, August
8. Joux A, Peyrin T (2007) Hash functions and the (Amplified) boomerang attack. In: Menezes A (ed) Advances in Cryptology – CRYPTO'07. Lecture notes in computer science, vol 4622. Springer, Berlin, pp 244–263
9. National Institute of Standards and Technology (NIST) (1995) FIPS Publication 180–1. Secure Hash Standard
10. National Institute of Standards and Technology (NIST) (2002) FIPS Publication 180–2. Secure Hash Standard
11. Saarinen M-JO (2003) Cryptanalysis of block ciphers based on SHA-1 and MD5. In: Johansson T (ed) Fast software encryption – FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 36–44
12. Sanadhya SK, Sarkar P (2008) New collision attacks against Up to 24-Step SHA-2. In: Chowdhury DR, Rijmen V, Das A (eds) INDOCRYPT'08. Lecture notes in computer science, vol 5365. Springer, Berlin, pp 91–103
13. Wang X, Yin Y-L, Yu H (2005) Collision Search Attacks on SHA-1. Unpublished manuscript

S

Shamir's Threshold Scheme

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Related Concepts

►Secret Sharing Schemes; ►Threshold Cryptography

Definition

A technique by means of polynomials to let k people recover a secret, while any combination of up to $k-1$ people does not even have partial information about that secret.

Background

In Ref. [1], A. Shamir proposed an elegant "polynomial" construction of a perfect ►*threshold schemes*. An (n,k)-threshold scheme is a particular case of a ►*secret sharing scheme* when any set of k or more participants can recover the secret exactly while any set of less than k particiants gains no additional, that is, a posteriori, information about the secret. Such threshold schemes are called *perfect* and they were constructed in Refs. [1, 2]. Shamir's construction is the following.

Theory

Assume that the set S_0 of secrets is some ►*finite field* $GF(q)$ of q elements (hence q should be prime power) and that the number of participants of SSS $n < q$. The dealer chooses n different nonzero elements (points) $x_1, \ldots, x_n \in GF(q)$, which are publicly known. To distribute a secret s_0, the dealer generates randomly coefficients $g_1, \ldots, g_{k-1} \in GF(q)$, forms the polynomial $g(x) = s_0 + g_1 x \cdots + g_{k-1}x^{k-1}$ of degree less than k, and sends to the ith participant the ►*share* $s_i = g(x_i)$. Clearly, any k participants can recover the whole polynomial $g(x)$ and, in particular, its zero coefficient (or $g(0)$), since any polynomial of degree l is uniquely determined by its values in $l+1$ points and Lagrange interpolation formula shows how to determine it. On the other hand, the point 0 can be considered as an "evaluation point" x_0, corresponding to the dealer, since $s_0 = g(0)$. Then the above consideration shows that for any given shares $s_1 = g(x_1), \ldots, s_{k-1} = g(x_{k-1})$ all possible values of s_0 are equally probable, hence the scheme is perfect.

For some applications, it is convenient to have the maximal possible number n of participants equal to q, especially for $q = 2^m$. For Shamir's scheme $n < q$ but the following simple modification allows to have $n = q$. Namely, the dealer generates a random polynomial of the form $f(x) = f_0 + f_1 x \cdots + f_{k-2}x^{k-2} + s_0 x^{k-1}$ and distribute shares $s_i = f(x_i)$, where the x_i are different but not necessary nonzero elements of $GF(q)$. The perfectness of this scheme can be proved either directly (along the line of the above proof by considering the polynomial $h(x) = f(x) - s_0 x^{k-1}$ of degree at most $k-2$), or as an application of established in Ref. [3] the relationship between perfect (n,k)-threshold schemes and $(n+1,k)$ Reed–Solomon codes (►*cyclic codes*), since the above construction is equivalent to so-called 2-lengthening of Reed–Solomon codes.

Recommended Reading

1. Shamir A (1979) How to share a secret. Commun ACM 22(1): 612–613
2. Blakley R (1979) Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 national computer conference, New York, vol 48, pp 313–317
3. McEliece RJ, Sarwate DV (1981) On secret sharing and Reed-Solomon codes. Commun ACM 24:583–584

Shamir's Trick

►Simultaneous Exponentiation

Shannon's Maxim

►Kerckhoffs' Principle

Shannon's Model

Henk C. A. van Tilborg
Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Synonyms

Information theoretic model

Related Concepts

►Information Theory; ►Symmetric Cryptography; ►Secret Key Cryptography

Definition

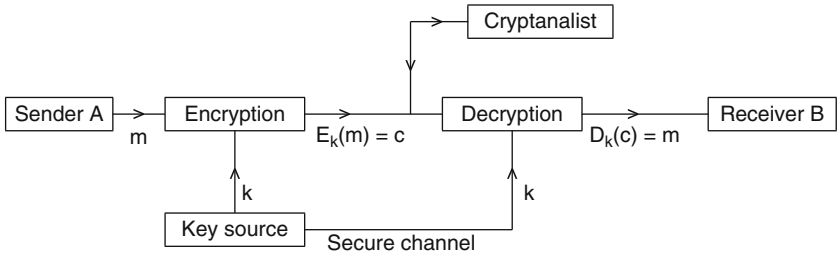
Shannon's model for secure communication is a mathematical way to formalize what the underlying principles of conventional cryptography are.

Background

Claude Shannon (1949) was the first to give a formal definition of what ►symmetric cryptosystems are. Of course, cryptosystems of all kinds have been around for at least two thousand years (►Caesar cipher), but a precise mathematical description of what is now called a symmetric cryptosystem had never been given before.

Theory

In Shannon's description (see Fig. 1), a sender A (often called Alice) wants to send message m to a receiver B (who



Shannon's Model. Fig. 1 Symmetric cryptography

is called Bob). The message is called a *plaintext* and is taken from a finite set, called plaintext space $\mathcal{M}$. Of course, Alice may want to send more messages.

Since the transmission channel is insecure (a person called Eve is also connected to the channel), Alice applies a mapping E_k to m. The result c is called the *ciphertext* and is an element of a set $\mathcal{C}$, the ciphertext space. The mapping E_k is called the *encryption* function. It is c that Alice sends to Bob and so it will be c that is intercepted by Eve.

Clearly, the encryption function E_k must be a one-to-one mapping, since Bob must be able to retrieve the plaintext/message m from the ciphertext c by means of the *decryption* function D_k. In formula: $D_k(c) = m$.

Since more people may want to use the same cryptosystem and since Alice and Bob do not want to use the same mapping too long for security reasons, their function is taken from a large set $\mathcal{E}$ of one-to-one mappings from $\mathcal{M}$ to $\mathcal{C}$. It is for this reason that the encryption and decryption functions depend on a quantity that is denoted by k. This k is called the ►key and is taken from the so-called *key-space* $\mathcal{K}$.

It is the set labeled encryption functions $\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$ that describes the cryptosystem. Quite clearly Alice and Bob must use the same key k. To this end, they use a *secure channel*, a communication line without any eavesdroppers. A possibility is that Alice and Bob agreed beforehand on the key, another possibility is that one has send the key by means of a courier to the other. Nowadays ►public key cryptography is often used for this purpose.

Normally, the same cryptosystem $\mathcal{E}$ will be used for a long time and by many people, so it is reasonable to assume that $\mathcal{E}$ is also known to the cryptanalist. It is the frequent changing of the key that has to provide the security of the data. This principle was already clearly stated by the Dutchman Auguste Kerckhoff (►maxims) in the 19-th century.

Often $\mathcal{M} = \mathcal{C}$. Ideally, the number of keys under which a particular plaintext is mapped to a particular ciphertext is the same, independent of the choice of the plaintext and ciphertext. In that case the ciphertext does not give any information about the plaintext (►information theory) and the system is called *unconditionally secure* (►Vernam cipher).

The cryptanalist who is connected to the transmission line can be:

passive (eavesdropping): The cryptanalist tries to find m (or even better k) from c.

active (tampering): The cryptanalist tries to actively manipulate the data that are being transmitted. For instance, Eve alters a transmitted ciphertext or retransmits it.

Applications

Shannon's model is very important from a theoretical perspective, but has limited value in practice. The reason is that Shannon's assumes that the adversary (Eve) has unlimited computing power. In reality, this is not the case. It is this insight that makes public key cryptography possible. Similarly, for most symmetric cryptosystems a single known plaintext-ciphertext pair of the same length as the key uniquely determines that key, according to Shannon, but finding that key will often be infeasible.

Recommended Reading

1. Shannon CE (1949) Communication Theory and Secrecy Systems. Bell Syst Tech J 28:656–715

Share

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Related Concepts

►Secret Sharing Schemes

Definition

Share is a portion of information distributed by a ►secret sharing scheme (SSS) to a given user. In the standard definition of SSS, shares are distributed via secure, private channels in such a way that each participant only knows his own share [1, 2]. We note that it is also possible to organize SSS in case of public channels [3].

Recommended Reading

1. Shamir A (1979) How to share a secret. Commun ACM 22(1): 612–613
2. Blakley R (1979) Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 national computer conference, vol 48. AFIPS Press, New York, pp 313–317
3. Beimel A, Chor B (1998) Secret sharing with public reconstruction. IEEE Trans Inform Theory 44(5):1887–1896

Shortest Vector Problem

Daniele Micciancio
Department of Computer Science & Engineering, University of California, San Diego, CA

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice Reduction; ►Lattice-Based Cryptography; ►NTRU

Definition

Given k linearly independent (typically integer) vectors $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ in n-dimensional Euclidean space $\mathbb{R}^n$, the *Shortest Vector Problem* (SVP) asks to find a nonzero *integer* linear combination $\mathbf{Bx} = \sum_{i=1}^{k} \mathbf{b}_i x_i$ (with $\mathbf{x} \in \mathbb{Z}^k \backslash \{\mathbf{0}\}$) such that the norm $\|\mathbf{Bx}\|$ is as small as possible. The problem can be defined with respect to any norm, but the Euclidean norm $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}$ is the most common. The set of all integer linear combinations $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx}: \mathbf{x} \in \mathbb{Z}^k\}$ is a ►lattice. So, SVP can be concisely defined as the problem of finding the shortest nonzero vector in the lattice represented by $\mathbf{B}$. (Refer the entry ►Lattice for general background about lattices, and related concepts.)

The problem can be relaxed in various ways, e.g., by asking just for the length (denoted λ_1) of the shortest nonzero lattice vector (without actually finding a lattice vector of length λ_1), or by allowing approximate solutions, i.e., nonzero lattice vectors of length at most $g \cdot \lambda_1$ for some approximation factor $g \geq 1$, which typically depends on the lattice dimension. In ►Computational Complexity and ►Lattice-Based Cryptography, the problem is most commonly encountered in its *length estimation* variant, which combines these two relaxations, and simply asks to approximate the value of λ_1 within a factor g. The decision problem (denoted GapSVP_g) corresponding to this task is: given a lattice basis $\mathbf{B}$ and a value d, determine if $\lambda_1(\mathbf{B}) \leq d$ or $\lambda_1(\mathbf{B}) > g \cdot d$. (When $d < \lambda_1(\mathbf{B}) \leq dg$, any answer is allowed.) Notice that the length λ_1 of the shortest nonzero lattice vector equals the minimum distance between any two distinct lattice points. For this reason, GapSVP is often referred to as the (approximate) minimum distance problem.

Background

SVP is the most famous and widely studied computational problem on lattices. In mathematics, the problem has been studied (in the equivalent language of quadratic forms) since the 19th century, mostly in connection to problems in number theory, and an efficient algorithm to solve it in two-dimensional lattices was already known to Lagrange and Gauss. However, no *efficient* approximation algorithm for SVP in arbitrary high-dimensional lattices was known until the early 1980s. In computer science and cryptography, the problem received a substantial amount of attention after the development of the LLL ►Lattice Reduction algorithm [4] in 1982 because of its many algorithmic applications, most notably in cryptanalysis, and more recently in connection with the development of ►Lattice-Based Cryptography.

Theory

A cornerstone mathematical result about SVP is *Minkowski's first theorem*, which states that the shortest nonzero vector in a lattice $\mathcal{L}$ has length at most $\gamma_n \cdot \det(\mathcal{L})^{1/n}$, where n is the lattice dimension, $\det(\mathcal{L})$ is the determinant of the lattice, and $\gamma_n = \Theta(\sqrt{n})$ is a proportionality factor (called Hermite's constant) that depends only on n (Refer the entry ►Lattice for a definition of *determinant*). The upper bound provided by Minkowski's theorem is tight, i.e., there are lattices such that the shortest nonzero vector has length $\gamma_n \cdot \det(\mathcal{L})^{1/n}$. However, general lattices may contain vectors much shorter than that. Moreover, Minkowski's theorem only proves that short vectors exist, i.e., it does not give an efficient algorithmic procedure to find such vectors.

The LLL lattice reduction algorithm ([4], ►Lattice Reduction) can be used to approximate SVP within a factor $g = O((2/\sqrt{3})^n)$, where n is the dimension of the lattice. Smaller (slightly ►subexponential) approximation factors can be achieved in ►polynomial time using more complex algorithms like Schnorr's *Block Korkine–Zolotarev* reduction [6].

SVP is NP-hard even in its length estimation version (GapSVP_g). The strongest NP-hardness result known to date for GapSVP_g (and therefore for SVP) was proved by Khot [3], who showed that the problem is NP-hard (under randomized reductions) to approximate within any constant factor $g = O(1)$, independent of the dimension n. Stronger (but still subpolynomial in n) inapproximability results are known for SVP in the ℓ_∞ norm [1]. On the other hand, Goldreich and Goldwasser [2] have shown that (under standard complexity assumptions) SVP cannot be NP-hard to approximate within small polynomial factors $g = O(\sqrt{n/\log n})$.

Applications

The (conjectured) worst-case hardness of approximating GapSVP_g within certain polynomial factors $g(n) = n^c$ can be used as the basis for the construction of provably secure cryptographic functions (►Lattice-Based Cryptography). ►Lattice Reduction for applications of SVP in cryptanalysis.

Open Problems

The main open problems about SVP concern its algorithmic solvability: are there efficient (polynomial time) algorithms to approximate SVP within polynomial factors $g(n) = n^c$, for some (large) constant $c < \infty$? Is SVP NP-hard to approximate within polynomial factors $g(n) = n^\epsilon$, for some (small) constant $\epsilon > 0$?

Another outstanding open question is to prove the equivalence between the search (SVP) and length estimation (GapSVP) versions of the approximate shortest vector problem. Currently, the two problems are known to be equivalent only in trivial cases, e.g., when the two problems are either NP-hard or solvable in polynomial time.

For an introduction to the computational complexity of the shortest vector problem and other lattice problems, see [5].

Recommended Reading

1. Dinur I (2002) Approximating SVP_∞ to within almost-polynomial factors is NP-hard. Theor Comput Sci 285(1):55–71
2. Goldreich O, Goldwasser S (2000) On the limits of nonapproximability of lattice problems. J Comput Syst Sci 60(3):540–563
3. Khot S (2005) Hardness of approximating the shortest vector problem in lattices. J ACM 52(5):789–808
4. Lenstra AK, Lenstra HW Jr, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261:513–534
5. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer International Series in Engineering and Computer Science, vol 671. Kluwer Academic Publishers, Boston
6. Schnorr C-P (1987) A hierarchy of polynomial time lattice basis reduction algorithms. Theor Comput Sci 53(2–3):201–224

Shrinking Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI, CNRS/Lab-STICC and Telecom Bretagne, Brest cedex 3, France

Related Concepts

►Clock-Controlled Generator; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

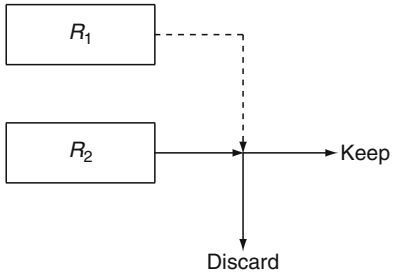
The shrinking generator is a ►clock-controlled generator that has been proposed in 1993 [1]. It is based on two ►Linear Feedback Shift Registers (LFSRs), say R_1 and R_2. The principle, as depicted in Fig. 1, is that R_1's output will decimate R_2's output. At each step, both are clocked; if R_1 outputs a 1, then R_2's output bit is included in the keystream, else (if R_1 outputs a 0) R_2's output bit is discarded.

Example

An example is provided in Fig. 2.

Theory

The inventors discussed some security points in their paper. A discussion about the implementation and the use of a buffer (in order to avoid the irregular rate of the output) is presented in [2] and [3]. Since the seminal survey of techniques for the cryptanalysis of clock-controlled generators [4], a lot of papers discussed (Fast) Correlation Attacks. Among the more recent papers one can cite [5–14]. But, according to these attacks, and if properly implemented, the shrinking generator remains resistant to practical cryptanalysis because these attacks cannot be considered as efficient when the LFSRs are too long for exhaustive search. Recently, following [15–19] proposed a



Shrinking Generator. Fig. 1 The shrinking generator

R_1		R_2		
State	Output	State	Output	Output
010		0101		
001	0	1010	1	
100	1	1101	0	0
110	0	0110	1	
111	0	0011	0	
011	1	1001	1	1
101	1	0100	1	1
010	1	0010	0	0
001	0	0001	0	
100	1	1000	1	1
110	0	1100	0	
111	0	1110	0	
011	1	1111	0	0
101	1	0111	1	1
...	...	...	...	...

Shrinking Generator. Fig. 2 R_1 has length three, and feedback relation $s_{t+1} = s_t + s_{t-2}$; R_2 has length four, and feedback relation $s_{t+1} = s_t + s_{t-3}$. The first row shows the initialization of the registers. The internal states are of the form $s_t s_{t-1} s_{t-2}$ or $s_t s_{t-1} s_{t-2} s_{t-3}$

new attack based on the so-called *edit/Levenshtein distance*, providing an attack which complexity is in $2^{L/2}$, the exhaustive search being in 2^L.

Recommended Reading

1. Coppersmith D, Krawczyk H, Mansour Y (1994) The shrinking generator. Advances in Cryptology – Crypto'93. Lecture notes in computer science, vol 773. Springer, pp 22–39
2. Kessler I, Krawczyk H (1995) Minimum buffer length and clock rate for the shrinking generator cryptosystem. IBM Research Report RC 19938, IBM T.J. Watson Research Center, Yorktown Heights, New York
3. Krawczyk H (1994) The shrinking generator: some practical considerations. Fast Software Encryption. Lecture notes in computer science, vol 809. Springer, pp 45–46
4. Gollmann D (1994) Cryptanalysis of clock-controlled shift registers. Fast Software Encryption. Lecture notes in computer science, vol 809, Springer, pp 121–126
5. Golic JD, O'Connor L (1995) Embedding and probabilistic correlation attacks von clock-controlled shift registers. Advances in Cryptology – Eurocrypt'94. Lecture notes in computer science, vol 950. Springer, pp 230–243
6. Golic JD (1995) Intrinsic statistical weakness of keystream generators. Advances in Cryptology – Asiacrypt'94. Lecture notes in computer science. Springer, pp 91–103
7. Golic JD (1995) Towards fast correlation attacks on irregularly clocked shift registers. Advances in Cryptology – Eurocrypt'95. Lecture notes in computer science, vol 921. Springer, pp 248–262
8. Jabri A, Kh. Al (1996) Shrinking generators and statistical leakage. Comput Math Appl 32(4):33–39, Elsevier Science
9. Johansson T (1998) Reduced complexity correlation attacks on two clock-controlled generators. Advances in Cryptology – Asiacrypt'98. Lecture notes in computer science, vol 1514. Springer, pp 342–356
10. Kholosha A (2001) Clock-controlled shift registers and generalized Geffe key-stream generator. Progress in Cryptology – Indocrypt'01. Lecture notes in computer science, vol 2247. Springer, pp 287–296
11. Kanso A (2003) Clock-controlled shrinking generator of feedback shift registers. AISP 2003. Lecture notes in computer science, vol 2727. Springer, pp 443–451
12. Daemen J, Van Assche G (2006) Distinguishing stream ciphers with convolutional filters. SCN 2006. Lecture notes in computer science, vol 4116. Springer, pp 257–270
13. Gomulkiewicz M, Kutylowski M, Wlaz P (2006) Fault jumping attacks against shrinking generator. Complexity of Boolean Functions, N. 06111, Dagstuhl Seminar Proceedings
14. Zhang B, Wu H, Feng D, Bao F (2005) A fast correlation attack on the shrinking generator. CT-RSA 2005. Lecture notes in computer science, vol 3376. Springer, pp 72–86
15. Golic JD, Mihaljevic MJ (1991) A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance. J Cryptol 3(3):201–212
16. Golic JD, Petrovic S (1993) A generalized correlation attack with a probabilistic constrained edit distance. Advances in Cryptology – Eurocrypt'92. Lecture notes in computer science, vol 658. Springer, pp 472–476
17. Petrovic S, Fúster A (2004) Clock control sequence reconstruction in the ciphertext only attack scenario. ICICS 2004. Lecture notes in computer science, vol 3269. Springer, pp 427–439
18. Caballero-Gil P, Fúster-Sabater A (2005) Improvement of the edit distance attack to clock-controlled LFSR-based stream ciphers. EUROCAST 2005. Lecture notes in computer science, vol 3643. Springer, pp 355–364
19. Caballero-Gil P, Fúster-Sabater A (2009) A simple attack on some clock-controlled generators. Comput Math Appl 58(1):179–188, Elsevier Science

Side-Channel Analysis

Marc Joye[1], Francis Olivier[2]
[1]Technicolor Security & Content Protection Labs, Cesson-Sévigné Cedex, France
[2]Department of Security Technology, Gemplus Card International, France

Introduction

Electronic devices have to comply with consumption constraints especially on autonomous equipments, like mobile phones. Power analysis has been included into most certification processes regarding products dealing with information security such as smart cards.

The electrical consumption of any electronic device can be measured with a resistor inserted between the ground

or Vcc pins and the actual ground in order to transform the supplied current into a voltage easily monitored with an oscilloscope.

Within a microcontroller the peripherals consume differently. For instance, writing into nonvolatile memory requires more energy than reading. Certain chips for smart cards enclose a crypto-processor, that is, a particular device dedicated to specific cryptographic operations, which generally entails a consumption increase. The consumption trace of a program running inside a microcontroller or a microprocessor is full of information. The signal analysis may disclose lots of things about the used resources or about the process itself. This illustrates the notion of *side channel* as a source of additional information.

Basically a power consumption trace exhibits large scale patterns most often related to the structure of the executed code. The picture below (Fig. 1) shows the power trace of a smart card chip ciphering a message with the Advanced Encryption Standard (AES). The ten rounds are easily recognized with nine almost regular patterns first followed by a shorter one.

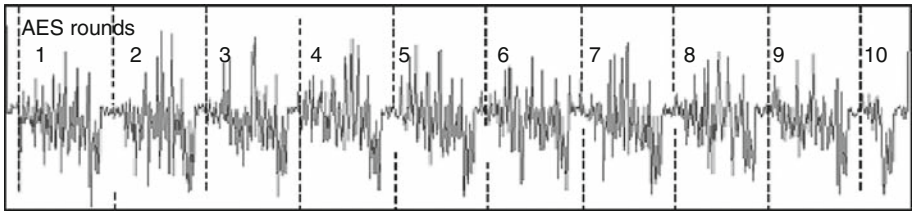
Zooming into a power signal exhibits a local behavior in close relationship with the silicon technology. At the cycle scale, the consumption curve looks roughly like a capacitive charge and discharge response.

A careful study of several traces of a same code with various input data shows certain locations where power trace patterns have different heights. The concerned cycles indicate some data dependence also called *information leakage*. They may be magnified by a variance analysis over a large number of executions with random data. For instance, by ciphering many random plaintexts with a secret-key algorithm, it is possible to distinguish the areas sensitive to input messages from the constant areas that correspond to the key schedule (Fig. 2).

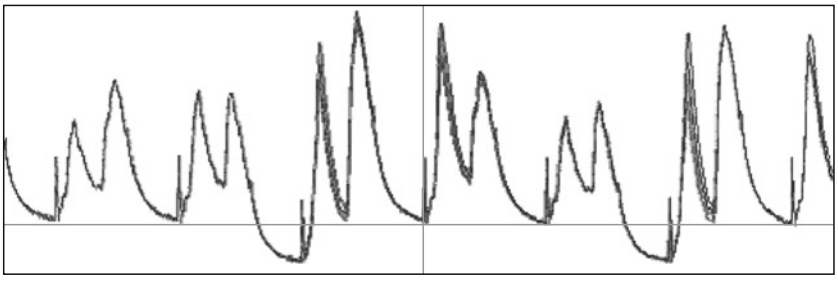
Information Leakage Model

The characterization of data leakage (namely, finding the relationships between the data and the variability of consumption) has been investigated by several researchers. The most common model consists in correlating these variations to the Hamming weight of the handled data, that is, the number of nonzero bits. Such a model is valid { } for a large number of devices. However, it can be considered as a special case of the transition model, which assumes that the energy is consumed according to the number of bits switched for going from one state to the next one. This behavior is represented by the Hamming distance between the data and some *a priori* unknown constant, that is, the Hamming weight of the data XOR-ed with this constant.

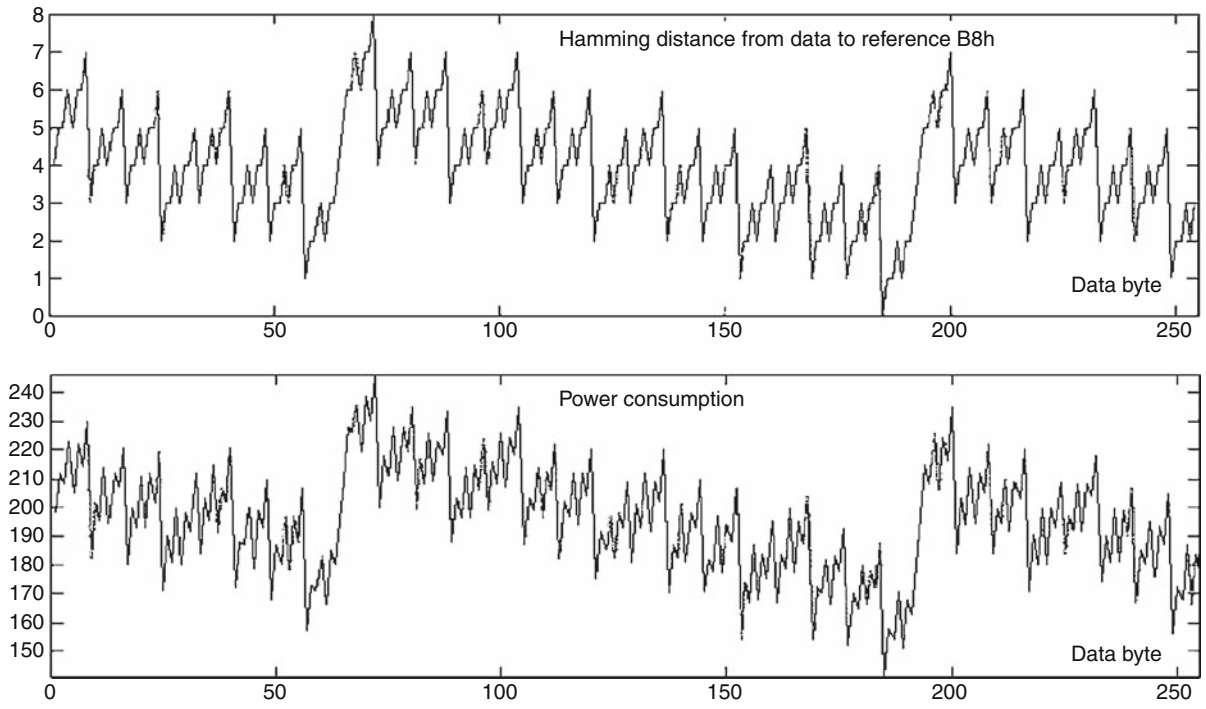
As shown in the next picture (Fig. 3), for an 8-bit microcontroller, the transition model may seem rough but it suffices to explain many situations, provided that the reference constant state is known. In most microprocessors this state is either an address or an operating code. Each of them has a specific binary representation and therefore a different impact in the power consumption: this is why each cycle pattern is most often different from its neighbors.



Side-Channel Analysis. Fig. 1 AES power trace



Side-Channel Analysis. Fig. 2 Information leakage



Side-Channel Analysis. Fig. 3 Transition model

Some technologies systematically go through a clear "all-zeros" state that explains the simpler Hamming-weight model.

Statistical Analyses

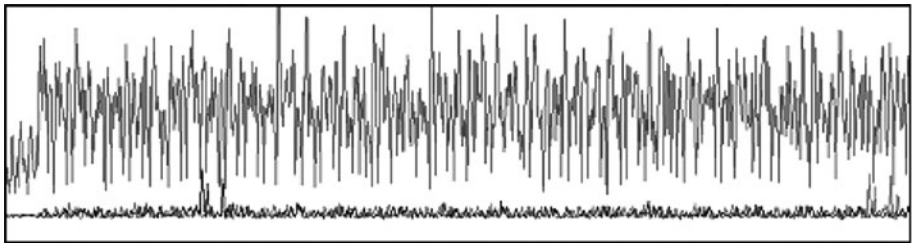
With information leakage models in mind, it is possible to design statistical methods in order to analyze the data leakage. They require a large amount of power traces assigned to many executions of the same code with varying data, generally at random, and make use of statistical estimators such as averages, variances, and correlations. The most famous method is due to Paul Kocher et al. and is called ►Differential Power Analysis (DPA).

Basically the purpose of DPA is to magnify the effect of a single bit inside a machine word. Suppose that a random word in a Ω-bit processor is known and uniformly distributed. Suppose further that the associated power consumption obeys the Hamming-weight model. On average the Hamming weight of this word is $\Omega/2$. Given N words, two populations can be distinguished according to an arbitrary selection bit: the first population, $\mathcal{S}_0$, is the set of t words whose selection bit is 0 and { }; the second population, $\mathcal{S}_1$, is the set of $N-t$ words whose selection bit is 1. On average, the words of set $\mathcal{S}_0$ will have a Hamming weight of $(\Omega-1)/2$ whereas the words of set $\mathcal{S}_1$ will have a Hamming weight of $(\Omega+1)/2$. The same bias can be seen through the corresponding power consumption traces since it is supposed to be correlated with the Hamming weight of the data. Let C_0 and C_1 respectively denote the averaged power consumption traces of the blue curve sets $\mathcal{S}_0$ and $\mathcal{S}_1$. The *DPA trace* is defined as the difference $C_0 - C_1$.

The resulting DPA curve has the property of erecting bias peaks at moments when the selection bit is handled. It looks like noise everywhere else: indeed, the constant components of the signal are cancelled by the subtraction whereas dynamic ones are faded by averaging, because they are not coherent with the selection bit.

This approach is very generic and applies to many situations. It works similarly with the transition model. Of course the weight of a single selection bit is relatively more important in processors with short words like 8-bit chips. If the machine word is larger, the same DPA bias can be obtained by increasing the number of trials (Fig. 4).

A first application of DPA is called *bit tracing*. It is a useful reverse engineering tool for monitoring a predictable bit during the course of a process. In principle a DPA peak rises up each time it is processed. This brings a lot of information about an algorithm implementation. To achieve the same goal Paul Fahn and Peter Pearson proposed another statistical approach called *Inferential Power Analysis* (IPA). The bits are inferred from the deviation between a single



Side-Channel Analysis. Fig. 4 Bit tracing (upper curve: power consumption of a single execution; two lower curves: DPA curves respectively tracing the first and last data bit of a targeted process)

trace and an average trace possibly resulting from the same execution: for instance, the average trace of a DES round (►Data Encryption Standard) can be computed over its sixteen instances taken from a single execution. IPA does not require the knowledge of the random data to make a prediction on a bit value. But as counterpart it is less easy to implement and the interpretation is less obvious.

After Paul Kocher, Thomas Messerges et al. have proposed to extend DPA by considering multiple selection bits in order to increase the signal to noise ratio (SNR). If the whole machine word is taken into account, a global approach consists in considering the transition model as suggested by Jean-Sébastien Coron et al.

From Power Analysis to Power Attacks

Obviously, if the power consumption is sensitive to the executed code or handled data, critical information may leak through power analysis. This section explains how to turn a side-channel analysis into an attack.

SPA-Type Attacks

A first type of power attacks is based on *Simple Power Analysis* (SPA). For example, when applied to an *unprotected* implementation of an ►RSA public key encryption scheme, such an attack may recover the whole private key (i.e., signing or decryption key) from a single power trace.

Suppose that a private RSA exponentiation, $y = x^d \bmod n$ (►Modular Arithmetic), is carried out with the square-and-multiply algorithm (refer also ►Exponentiation Algorithms) (Fig. 6). This algorithm processes the exponent bits from left to right. At each step there is a squaring, and when the processed bit is 1 there is an additional multiplication. A straightforward (i.e., unprotected) implementation of the square-and-multiply algorithm is given in Fig. 5.

The corresponding power curve exhibits a sequence of consumption patterns among which some have a low level and some have a high level. These calculation units are assigned to a crypto-processor handling n-bit arithmetic.

$k \leftarrow$ bitsize(d)
$y \leftarrow x$
for $i = k - 2$ **downto** 0 **do**
  $y \leftarrow y^2 \pmod n$
  if (bit i of d is 1) **then** $y \leftarrow y \cdot x \pmod n$
endfor
return y

Side-Channel Analysis. Fig. 5 Square-and-multiply exponentiation algorithm

Knowing that a low level corresponds to a squaring and that a high level corresponds to a multiplication, it is fairly easy to read the exponent value from the power trace:

- A low-level pattern followed by another low-level pattern indicates that the exponent bit is 0
- A low-level pattern followed by a high-level pattern indicates that the exponent bit is 1

This previous picture also illustrates why the Hamming weight of exponent d can be disclosed by a timing measurement.

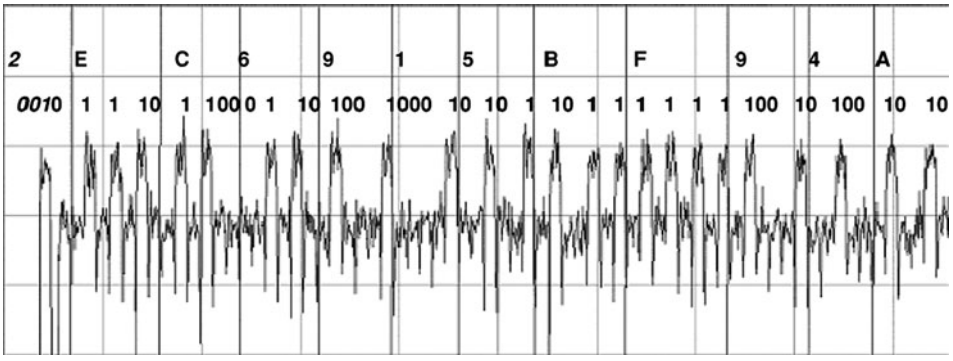
DPA-Type Attacks

Historically, DPA-type attacks – that is, power attacks based on *Differential Power Analysis* (DPA) – were presented as a means to retrieve the bits of a DES key.

At the first round of DES, the output nibble of the ith S-box $(1 \le i \le 8)$ can be written as $S_i(M \oplus K)$ where

- M is made of 6 bits constructed from the input message after IP- and E-permutations: it has to be chosen at random but is perfectly known and predictable.
- K is a 6-bit sub-key derived from the key scheduling.

Rising up a DPA bias would require the knowledge of the output nibble. As K is unknown to the adversary this is not possible. But sub-key K can be easily exhausted as



Side-Channel Analysis. Fig. 6 SPA trace of the basic square-and-multiply algorithm

it can take only $2^6 = 64$ possible values. Therefore the procedure consists in reiterating the following process for $0 \leq \hat{K} \leq 63$:

1. Form sets $\mathcal{S}_0 = \{M \mid g(\text{S-box}_i(M \oplus \hat{K})) = 0\}$ and $\mathcal{S}_1 = \{M \mid g(\text{S-box}_i(M \oplus \hat{K})) = 1\}$ where selection function g returns the value of a given bit in the output nibble.
2. Compute the corresponding DPA curve.

In principle the bias peak should be maximized when the guess $\hat{K}$ is equal to the real sub-key K. Then inverting the key schedule permutation leads to the value of 6 key bits. In other words the DPA operator is used to validate sub-key hypotheses. The same procedure applies to the 7 other S-boxes of the DES. Therefore, the whole procedure yields $8 \times 6 = 48$ key bits. The 8 remaining key bits can be recovered either by exhaustion or by conducting a similar attack on the second round.

The main feature of a DPA-type attack resides in its genericity. Indeed it can be adapted to many cryptographic routines as soon as varying and known data are combined with secret data through logical or arithmetic operations.

A similar attack can be mounted against the first round of ►Rijndael/AES; the difference being that there are 16 byte-wise bijective substitutions and therefore 256 guesses for each. Finally, note that DPA-type attacks are not limited to symmetric algorithms; they also apply to certain (implementations of) asymmetric algorithms, albeit in a less direct manner.

Other Attacks

Among the other statistical attacks, IPA is more difficult and less efficient. Its purpose is to retrieve key bits without knowing the processed data. It proceeds by comparing the power trace of a DES round with an average power trace computed, for instance, over the 16 rounds. In principle, key bits could be inferred this way because the differential curve should magnify the bits deviation where they are manipulated (Fig. 7).

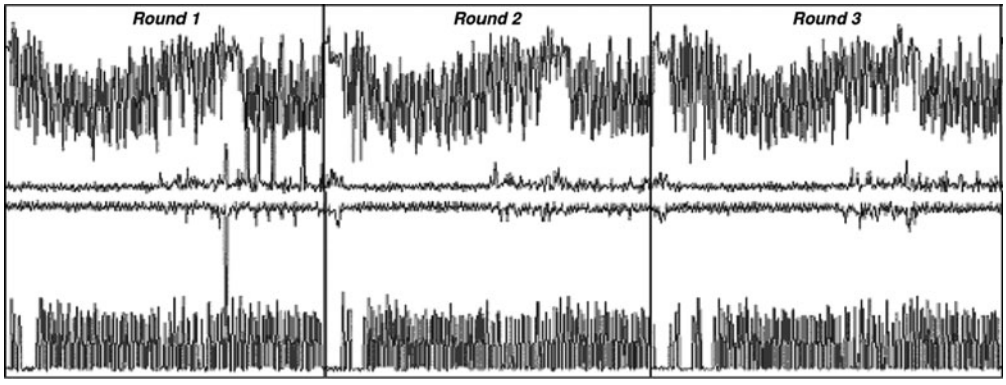
Dictionary (or template) attacks can be considered as a generalization of IPA to very comfortable but realistic situations. They have been widely studied in the field of smart cards where information on secret key or ►personal identification numbers (PIN) could potentially be extracted. They consist in building a complete dictionary of all possible secret values together with the corresponding side-channel behavior (e.g., power trace) when processed by the device (e.g., for authentication purpose). Then a secret value embedded in a twin device taken from the field can be retrieved by comparing its trace and the entries of the dictionary.

In practice, things do not happen that easily for statistical reasons and application restrictions. Only part of the secret is disclosed and the information leakage remains difficult to exploit fully.

Finally, in addition to power consumption, other side channels can be considered; possible sources of information leakage include running time or electromagnetic radiation.

Countermeasures

The aforementioned attacks have all been published during the second half of the 1990s. In view of this new threat the manufacturers of cryptographic tokens have designed a large set of dedicated countermeasures especially to thwart statistical attacks like DPA. All the related research activity has now resulted in tamper-resistant devices widely available in the market. It has given rise to the new concept of "secure implementation," which states that information leakage is not only due to the specification of an application



Side-Channel Analysis. Fig. 7 DPA trace of the three first rounds of DES (two upper (respectively lower) curves: power consumption curve of maxima (respectively minima) of a single execution and DPA curve of maxima (respectively minima))

(cryptographic processing or whatever), but also to the way it is implemented.

If information leaks through a physical side-channel there are two defensive strategies. The first consists in decorrelating the secret data from the side-channel. The second consists in decorrelating the side-channel from the secret data. The borderline between both is sometimes fuzzy but roughly speaking, the former is rather software oriented and intends to mask the data since they have to leak anyway, whereas the latter is more hardware oriented and intends to shut the side-channel physically in order to make the device tamper-resistant.

Chip manufacturers have introduced into their hardware designs many security features against power attacks. They are stricto sensu countermeasures since they aim at impeding the power measurement and make the recorded signal unworkable.

- Some countermeasures consist in blurring the signal using smoothing techniques, additive noise, or desynchronization effects. These countermeasures are poorly efficient against SPA working on broadscale traces. They are rather designed against statistical attacks. They may require some complementary circuits to generate parasitic components into the consumed current. Desynchronization aims at misaligning a set of power traces by the means of unstable clocking or the insertion of dummy cycles at random, making the statistical combination of several curves ineffective.
- Other countermeasures rather intend to decrease or cancel the signal at the source. Reduction is a natural consequence of the shrinking trend in the silicon industry that diminishes the power consumption of each elementary gate. More interesting (and expensive) is the emerging technology called "precharged dual rail logic" where each bit is represented by a double circuitry. At a given time a logical 0 is represented physically by a 01, and a logical 1 by 10. The transition to the next time unit goes through a physical 00 or 11 state so that the same amount of switching occurs whatever the subsequent state is. Consequently if both rails are perfectly balanced, the overall consumption of a microprocessor does not depend on the data anymore.

Software countermeasures enclose a large variety of techniques going from the application level to the most specific algorithmic tricks. One can classify them into three categories: application constraints, timing countermeasures, and data masking.

- Application constraints represent an obvious but often forgotten means to thwart statistical analyses. For instance, DPA requires known data with a high variability. An application wherein an input challenge (or an output cryptogram) would be strictly formatted, partially visible and constrained to vary within hard limits (like a counter) would resist DPA fairly well.
- Timing countermeasures mean the usage of empirical programming tricks in order to tune the time progress of a process. A critical instruction may have its execution instant randomized by software: if it never occurs at the same time, statistical analysis becomes more difficult. Conversely other situations require the code to be executed in a constant time, in order to protect it from SPA or timing analysis. For instance, a conditional branch may be compensated with a piece of fake code with similar duration and electrical appearance.
- Data masking (also known as whitening or randomization) covers a large set of numerical techniques

designed by cryptographers and declined in various manners according to the algorithm they apply to. Their purpose is to prevent the data from being handled in clear and to disable any prediction regarding their behavior when seen through the side channel. For example, the modular exponentiation $y = x^d \bmod n$(as used in the ►RSA public key cryptosystem) can be evaluated as:

$$y = \{(x + r_1 n)^{d + r_2 \varphi(n)} \bmod r_3 n\} \bmod n$$

for randoms r_i and where φ denotes ►Euler totient function.

To illustrate how fuzzy the borderline between hardware and software countermeasures can be, authors have mentioned that, for instance, desynchronization can be implemented by hardware or software means. The same remark applies to data masking for which some manufacturers have designed dedicated hardware tokens or mechanisms such as bus encryption or wired fast implementations of symmetric algorithms.

The experience shows that combined countermeasures act in synergy and increase the complexity in a much larger proportion than the sum of both. For instance, the simple combination of desynchronization tricks and data masking makes DPA (or more sophisticated variants thereof) quite harmless. In the same way, new hardware designs resist the most state-of-the-art and best equipped experts.

Recommended Reading

1. Chari S, Charanjit S, Jutla JR, Rao PR (1999) Towards sound approaches to counteract power-analysis attacks. In: Wiener M (ed) Advances in cryptology – CRYPTO'99, Santa Barbara. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, pp 398–412
2. Coron J-S, Paul K, David N (2001) Statistics and secret leakage. In: Frankel Y (ed) Financial cryptography (FC 2000), Anguilla. Lecture Notes in Computer Science, vol 1962. Springer, Berlin, pp 157–173
3. Fahn PN, Peter KP (1999) IPA: a new class of power attacks. In: Koç ÇK, Paar C (eds) Cryptographic hardware and embedded systems (CHES'99), Worcester. Lecture Notes in Computer Science, vol 1717. Springer, Berlin, pp 173–186
4. Gandolfi K, Christophe M, Francis O (2001) Electromagnetic analysis: concrete results. In: Koç ÇK, Naccache D, Paar C (eds) Cryptographic hardware and embedded systems – CHES 2001, Paris. Lecture Notes in Computer Science, vol 2162. Springer, Berlin, pp 251–261
5. Kocher P (1996) Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96, Santa Barbara. Lecture Notes in Computer Science, vol 1109. Springer, Berlin, pp 104–113
6. Kocher P, Joshua J, Benjamin J (1999) Differential power analysis. In: Wiener M (ed) Advances in cryptology – CRYPTO'99, Santa Barbara. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, pp 388–397
7. Messerges TS (2000) Using second-order power analysis to attack DPA resistant software. In: Koç ÇK, Paar C (eds) Cryptographic hardware and embedded systems – CHES 2000, Worcester. Lecture Notes in Computer Science, vol 1965. Springer, Berlin, pp 238–251
8. Messerges TS, Ezzy AD, Robert HS (2002) Examining smart-card security under the threat of power analysis attacks. IEEE Trans Comput 51(5):541–552

Side-Channel Attacks

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Synonyms

Environmental attacks

Related Concepts

►Differential Power Analysis; ►Fault Attack; ►Tempest; ►Timing Attack; ►Side-Channel Analysis

Theory

Side-channel attacks or environmental attacks of cryptographic modules exploit characteristic information extracted from the implementation of the cryptographic primitives and protocols. This characteristic information can be extracted from timing, power consumption, or electromagnetic radiation features (►Tempest). Other forms of side-channel information can be a result of hardware or software faults, computational errors, and changes in frequency or temperature. Side-channel attacks make use of the characteristics of the hardware and software elements as well as the implementation structure of the cryptographic primitive. Therefore, in contrast to analyzing the mathematical structure and properties of the cryptographic primitives only, ►side-channel analysis also includes the implementation. Some implementations are more vulnerable to specific side-channel attacks than others. Examples of attacks based on side-channel analysis are differential power attacks examining power traces (►Differential Power Analysis), timing attacks measuring the amount of time used to complete cryptographic operations (►Timing Attack), and fault induction attacks exploiting errors in the computation process of cryptographic primitives (►Fault Attacks).

Side-Channel Leakage

►Cryptophthora

Sieving

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Function Field Sieve; ►Index Calculus; ►Number Field Sieve for Factorization; ►Number Field Sieve for the DLP; ►Quadratic Sieve; ►Sieving in Function Fields; ►Smoothness

Definition

Sieving refers to a process for selecting candidates for further processing among a set of elements, typically by employing arithmetic progressions to identify candidates to be filtered out.

Theory

When searching for elements in a set with a given property, it is often possible to improve efficiency by filtering out candidates based on simple tests first rather than applying a full test for the property to all elements. A "sieve" is a test that an element must pass to be considered further; "sieving" is the process of applying the tests.

For instance, in the *Sieve of Eratosthenes* (►Prime Number), candidate primes are selected from a range of integers by crossing off elements divisible by small primes 2, 3, 5, 7, 11, 13,... Crossing off every second, third, fifth, seventh element, and so on is generally faster than testing each element separately for divisibility by small primes. Furthermore, only the remaining candidates need to be subjected to a full and more expensive ►primality test.

Applications

In addition to the use of the speedup in primality testing, sieving is the first and major phase of the fastest general algorithms for ►integer factoring and for solving the ►discrete logarithm problem. Here, the candidates sought are those that are divisible only by small primes or their equivalent (►Smoothness and ►Factor Base). Specific examples of sieving are described further in the related concepts. Refer also ►TWIRL for a design for efficient sieving in hardware.

Sieving in Function Fields

Emmanuel Thome
INRIA Lorraine, Campus Scientifique, Villers-lès-Nancy Cedex, France

Related Concepts

►Discrete Logarithm Problem; ►Function Field Sieve; ►Hyperelliptic Curves

Definition

Function fields provide a mathematical setting which is very similar to ►number fields, at least as far as cryptographic applications are concerned. Number fields are algebraic extensions of the field of rational numbers, and analogously function fields (as considered in cryptographic applications) are algebraic extensions of the univariate rational function field over a finite field $\mathrm{GF}(q)$. To quite a large extent, the point of view of number fields remains valid in the function field setting, by having the polynomial ring $\mathrm{GF}(q)[x]$ play the role of integers. Function fields are also equipped with a rich structure originating from the corresponding algebraic curve. The definition polynomial of a function field F may be viewed as a bivariate polynomial over a finite field, and as such defines a curve $\mathcal{C}$ in the projective plane $\mathbb{P}^2(\mathrm{GF}(q))$. The Jacobian group J of the curve $\mathcal{C}$ over $\mathrm{GF}(q)$ is exactly the degree zero divisor class group $Cl^0(F)$ of the function field F (►Hyperelliptic Curves for related concepts).

In contrast with number fields, function fields benefit from the fact that several problems are much easier to handle in the polynomial ring $\mathrm{GF}(q)[x]$ than over the integers: factoring polynomials has polynomial complexity, counting ►irreducible polynomials below a given degree is easy. On the theoretical side, an analogue of the Riemann hypothesis has been proved for function fields, thus there exists no specific "under GRH" results in the context of function fields.

Applications

With regard to the concepts of ►smoothness and ►sieving, function fields enjoy properties similar to the case of number fields. When considering the function field as an extension of $\mathrm{GF}(q)[x]$, one can consider the factorization of the norm of a given function. Whether this norm is smooth or not occurs with probabilities which are similar to the probability of the corresponding event in number fields (smoothness of the norm, which is an integer). One is thus naturally led to consider how algorithms of the broad family of ►index-calculus attacks, e.g., the

►quadratic sieve and the ►number field sieve carry over to the context of function fields.

An early algorithm of this large family instantiated in the context of function fields and curves is the Adleman-De Marrais-Huang algorithm (1992), which can be viewed as a function field version of the Hafner-McCurley algorithm for computing class groups of imaginary quadratic fields. This algorithm is described as follows. After having selected an appropriately large set of prime divisors of bounded degree called the ►factor base, one tries to express random functions as sums of divisors of the factor base. This gives insight into the degree zero divisor class group as a finitely presented group. This forms a tool to solve the discrete logarithm problem, and unveil the cardinality and structure of the Jacobian group, in heuristic time $L_{q^g}(1/2) = \exp\left((c+o(1))\sqrt{g\log q}\sqrt{\log(g\log q)}\right)$ (►L-notation).

The process of sieving, known for long to be successful for the quadratic sieve algorithm, for example, can be adapted to the function field case. The first task is to quickly identify the set of functions (amongst a set fixed beforehand) which have a prescribed zero at a given place. The divisor of such functions contains the corresponding prime divisor in its support. It is possible to restate this identification problem as a simple linear algebra problem and thus enumerate all such functions efficiently. The sieving procedure updates then a sum of "contributions" arising for all potential prime divisors into an array of functions to consider. Functions totalling a large contribution at the end of this procedure are ►smooth and yield relations in the divisor class group.

The efficiency of sieving in function fields is largely dependent on the definition polynomial. Indeed, while the sieving algorithm makes it possible to concentrate on functions of relatively small degree, it is essential that the norm of these function remains acceptably small for a better ►smoothness probability. This is apparent in the ►Function Field Sieve algorithm for instance (Relevant Article), but also for discrete logarithm computation on curves where a suitable curve equation is the key to obtaining an analogue of the Adleman-De Marrais-Huang with the $L(1/2)$ complexity replaced by the faster $L(1/3)$. Such an algorithm has been recently proposed by Enge and Gaudry [3].

Recommended Reading

1. Adleman LM, Huang M-D (1999) Function field sieve methods for discrete logarithms over finite fields. Inform Comput 151(1):5–16
2. Adleman LM, De Marrais J, Huang M-D (1994) A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In: Adleman LM, Huang M-D (eds) 1st algorithmic number theory symposium (ANTS-I), Cornell University, 6–9 May 1994. Lecture notes in computer science, vol 877. Springer, Berlin, pp 28–40
3. Enge A, Gaudry P (2007) An $L(1/3+\epsilon)$ algorithm for the discrete logarithm problem for low degree curves. In: Naor M (ed) Advances in cryptology – EUROCRYPT 2007. Lecture notes in computer science, vol 4515. Springer, Berlin, pp 379–393
4. Flassenberg R, Paulus S (1999) Sieving in function fields. Exp Math 8:339–349
5. Hafner JL, McCurley KS (1989) A rigorous subexponential algorithm for computation of class groups. J Am Math Soc 2(4): 837–850

Σ-Protocols

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Interactive Proof; ►Zero-knowledge

Definition

A Σ-protocol is a particular type of three-message interactive zero-knowledge proof (or argument) of knowledge. The prover sends the first message, followed by a random challenge from the verifier, and concluded with a response from the prover. The protocol is required to satisfy perfect completeness, special soundness, and special honest-verifier zero-knowledge.

Background

The term Σ-protocol was introduced by Cramer in his Ph.D. thesis [4] as a common abstraction of a broad class of interactive proofs. Originally, Σ-protocols were defined based on their form of a three-message interactive proof between a prover and a verifier, both modeled as probabilistic polynomial-time interactive Turing machines. The prover sends the first and third message, keeping the random bits used for computing these messages private, while the verifier sends the second message, which essentially consists of a number of publicly revealed random bits. The term Σ-protocol, however, has become synonymous with the particular type of protocol to which the well-known construction of [6] applies. The efficient zero-knowledge (or, witness-hiding) identification protocols due to Fiat-Shamir [7], Guillou-Quisquater [8], Schnorr [10], and Okamoto [9] are prominent instances of Σ-protocols. Many more instances and even classes of basic

Σ-protocols are known in the literature, covering all kinds of cryptographic (number-theoretic) settings.

Theory

A Σ-protocol is defined with respect to a given binary relation $R = \{(v; w)\}$. Relation R can generally be thought of as an NP-relation, which means that given instance v and *witness* w, it can be verified in time polynomial in the size of v whether $(v; w) \in R$ holds. In other words, the corresponding language $L_R = \{v : \exists_w (v; w) \in R\}$ is in *NP*, the well-known class of nondeterministic polynomial-time languages.

A Σ-protocol for a given relation R is a protocol between a prover P and a verifier V of the following form. The instance v is common input to P and V, while witness w is a private input to P. The prover basically starts by sending the verifier a *commitment* a to a privately chosen random value. Then the verifier sends the prover a random *challenge* c, to which the prover replies with the proper *response* r. The verifier checks the *conversation* $(a; c; r)$ against a deterministic polynomial-time predicate ϕ, and if this check succeeds $(a; c; r)$ is said to be *accepting*.

A Σ-protocol for relation R is required to satisfy the following three properties: (1) *Perfect completeness*, which means that if P and V follow the protocol, then V always accepts. (2) *Special soundness*, which says that for any $v \in L_R$ and any pair of accepting conversations $(a; c; r)$ and $(a; c'; r')$ with $c \neq c'$, one can efficiently extract a witness w such that $(v; w) \in R$. (3) *Special honest-verifier (perfect) zero-knowledge*, which means one can efficiently simulate conversations $(a; c; r)$ between honest P and V given any $v \in L_R$ and any fixed challenge c. The latter property can be stated more formally as the existence of a probabilistic polynomial-time *simulator* S which for given v and c produces conversations $(a; c; r)$ perfectly indistinguishable from conversations between honest P and V with common input v, where P uses any witness w with $(v; w) \in R$ and V uses challenge c. Perfect indistinguishability means that the probability distributions are identical.

A simple example is Schnorr's identification protocol. An instance $v = (g, p, y)$ consists of a generator g of an abstract group of prime order p and $y \in \langle g \rangle$, and the corresponding witness is $w = \log_g y$. The prover sends $a = g^u$ for $u \in_R \mathbb{Z}_p$ to the verifier. The verifier replies with a challenge, say $c \in_R \mathbb{Z}_p$. Finally, the prover sends the response $r = u + cw \bmod p$, and the verifier checks that $g^r = ay^c$ holds. Perfect completeness evidently holds. Special soundness holds because if $g^r = ay^c$ and $g^{r'} = ay^{c'}$ holds, then $\log_g y = (r - r')/(c - c')$, using $c \neq c'$, hence witness w can be computed from r, r', c, c'. Finally, special honest-verifier (perfect) zero-knowledge holds because the simulator may simply generate conversations $(a; c; r)$ for given challenge c by picking $r \in_R \mathbb{Z}_p$ and setting $a = g^r y^{-c}$.

If the challenges are chosen from an exponentially large set, the special soundness property implies that a Σ-protocol is a *proof of knowledge*. This ensures that a cheating prover cannot succeed with a significant probability of success unless it knows a valid witness for v. The zero-knowledge property demanded from a Σ-protocol is rather weak – covering honest verifiers only – but this ensures that Σ-protocols behave nicely under many types of composition [3, 6]. Moreover, there are several techniques to transform a Σ-protocol into a protocol withstanding arbitrarily cheating verifiers. For instance, the Fiat–Shamir technique [7] is to replace the randomly chosen challenge c with a cryptographic hash H applied to message a (and potentially further input values); this makes the entire protocol noninteractive and provably secure in the random oracle model. When applied to the identification protocols of [7, 8, 10], one gets the corresponding signature schemes (using the message to be signed as an input to H as well).

Applications

Routine use of Σ-protocols is made to provide efficient zero-knowledge proofs in many advanced cryptographic schemes. The zero-knowledge proofs are often used to ensure honest behavior of parties running a protocol. Using the basic techniques for monotone compositions (including AND/OR/threshold-compositions) of [6], for EQ-composition of [3], and further basic techniques (for instance, [2, 5]) succinct proofs for many relations (or statements) of interest can be rendered. Commonly, these proofs are made noninteractive using the Fiat–Shamir technique or generalizations thereof [1].

Recommended Reading

1. Abdalla M, An JH, Bellare M, Namprempre C (2002) From identification to signatures via the fiat-shamir transform: minimizing assumptions for security and forward-security. In: Advances in cryptology—EUROCRYPT '02, Amsterdam, 28 April–2 May 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 418–433
2. Brands S (1997) Rapid demonstration of linear relations connected by boolean operators. In: Advances in cryptology—EUROCRYPT '97, Konstanz, 11–15 May 1997. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 318–333
3. Chaum D, Pedersen TP (1993) Wallet databases with observers. In: Advances in cryptology—CRYPTO '92, Santa Barbara, 16–20 August 1992. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
4. Cramer R (1997) Modular design of secure yet practical cryptographic protocols. Ph. D thesis, Universiteit van Amsterdam, Netherlands

5. Cramer R, Damgård I (1998) Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In: Advances in cryptology—CRYPTO '98, Santa Barbara, 23–27 August 1998. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 424–441
6. Cramer R, Damgård I, Schoenmakers B (1994) Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in cryptology—CRYPTO '94, Santa Barbara, 21–25 August 1994. Lecture notes in computer science, vol 839. Springer, Berlin, pp 174–187
7. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Advances in cryptology—CRYPTO '86, Santa Barbara, 11–15 August 1986. Lecture notes in computer science, vol 263. Springer, New York, pp 186–194
8. Guillou LC, Quisquater J-J (1988) A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Advances in cryptology—EUROCRYPT '88, Davos, 25–27 May 1988. Lecture notes in computer science, vol 330. Springer, Berlin, pp 123–128
9. Okamoto T (1993) Provably secure and practical identification schemes and corresponding signature schemes. In: Advances in cryptology—CRYPTO '92, Santa Barbara, 16–20 August 1992. Lecture notes in computer science, vol 740. Springer, Berlin, pp 31–53
10. Schnorr CP (1991) Efficient signature generation by smart cards. J Cryptol 4(3):161–174

Signature Biometrics

Patrizio Campisi, Emanuele Maiorana, Alessandro Neri
Department of Applied Electronics, Università degli Studi Roma TRE, Rome, Italy

Related Concepts

►Handwriting

Definition

An handwritten signature represents the name of a person written by his/her own hand. Within the biometric framework, signature is a behavioral characteristic which is used in automatic user verification systems.

Background

Handwritten signature has a long history as a mean to verify the identity of a user for applications like document authentication, credit card transactions, cheque verification, attendance recording, and forensics. Although signature is still used to verify the identity of an individual by visual inspection, a constantly growing interest by both academia and industry has lead to the development of systems for automatic signature-based user verification. Nowadays, signature is one of the most accepted biometrics since it is perceived as a noninvasive and nonthreatening characteristics by the majority of the users being the act of signing part of everyday life. A signature can be easily acquired either using a classic ink pen or by means of specialized devices such as graphic tablet, pen-sensitive computer display, digital pens, or PDA. Within the biometric framework, signature is a behavioral characteristic since it can be influenced by physical and emotional conditions. Moreover, signature biometrics is not permanent, since it might vary along time. Therefore, different signature realizations, from the same user, can exhibit significant variability, which must be taken into account in the verification process. In addition, signature cannot be used by illiterate people thus not being a universally employable trait. Differently from the majority of other biometrics, signature can be reissued. In fact, if it is compromised, the user, with a certain degree of effort, can change his/her signature. Because of the wide social and economical impact of applications based on signature verification, in the last decades a huge effort has been devoted to research in this field. A review of the state of the art covering the literature up to 1993 can be found in [1, 2]. Survey papers quoting the more recent advances in signature recognition up to 2004 are given in [3, 4].

Theory

Automatic signature recognition is mainly used in the verification modality, i.e., in the process of confirming an identity claim through biometric comparisons. The use of signature biometrics in the identification modality, i.e., in the process of searching a database for a reference, matching a submitted biometric sample, and returning the corresponding identity, is neither practical nor very accurate.

Signature-based verification can be either *static* or *dynamic*. In the *static* mode, also referred to as *off-line*, only the spatio-luminance evolution of the signature, either acquired through a camera or an optical scanner, is available. In this case, some geometric characteristics of the signature image, namely height, width, and many others, can be extracted from the whole signature or from part of it. In the *dynamic* mode, also called *on-line*, the spatio-temporal evolution of the signature is acquired by means of a graphic tablet, a pen-sensitive computer display, a PDA, or digital pens. Since on-line signature verification involves dynamic features, which are much more difficult to forge than static ones, it is more suitable for user verification in legal and commercial transactions requiring high-security than off-line systems.

An on-line signature verification process is composed by the following steps:

- *Acquisition*: Digitizing tablets or PC touch screens can be used to acquire dynamic information. Specialized pens, which are able to provide spatial coordinates, pressure values, and pen inclination angles can also be used. The usual temporal sampling rate of these devices ranges from 100 to 200 Hz which is far beyond the maximum frequency of biomechanical sequences which is 30-40 Hz, thus fulfilling the requirements of the Nyquist theorem. The spatial sampling usually range from 100 to 300 dot per inch.
- *Preprocessing*: Once the signature has been acquired, some preprocessing is usually needed in order to prealign the signature, to normalize the signature dimensions, to localize the signature, to denoise the acquired data, to segment the signature, and so on. The preprocessing can be performed either in the spatial or in a transform domain like the Fourier, wavelet, Hadamard, or Hough domain, to cite a few.
- *Feature extraction*: In order to represent the signature itself, some features must be extracted. As widely accepted in the current literature, two different kinds of features can be considered: *parameters* and *functions*. The former refer to scalar values, derived from the acquired signature and then collected in a feature vector. Both dynamic information, like the number of pen-down events and the signature duration, and static information such as the signature height and width in different segments, can be extracted from the signature. Function features refer to on-line acquisitions where functions, like the position of the pen tip in the (x, y) plane, the pressure applied by the pen, the azimuth and the altitude angle of the pen with respect to the tablet, are acquired versus time and used to represent the signature. From this set of features, some others can be obtained like for example velocity, acceleration, curvature radius, tangential and centripetal acceleration, and trajectory angle versus time. However, not all the features have the same level of reliability. From an ideal point of view, a reliable feature should have values close enough for different realizations of a genuine signature, whereas far enough when they are extracted from forged signatures. In [5], an analysis of the reliability of *parameter* features has been carried out.
- *Matching*: The final step of the verification process is the matching stage which consists of measuring the similarity between the extracted features and the ones in the database. When parameters are employed as features, the similarity is evaluated by using distance measures such as Euclidean distance, Mahalanobis distance, etc. When functions are used as features, they can be used to characterize the underlying stochastic model used to represent the user's biometrics. The matching is then performed by verifying the similarity between the reference model, characterized in the enrollment stage, and the model characterized using the signature to verify. The most popular models employed in signature verification are the hidden Markov models (HMMs) [6, 7], which represent doubly embedded stochastic processes, composed by an underlying Markov chain whose states are not observable, and a set of stochastic processes which produce a sequence of available observations. The time functions representing a signature can be also directly matched using deformable distance measures such as dynamic time warping (DTW) [8–10], which finds the best nonlinear alignment between two vectors such that the overall distance between them is minimized.
- *Decision*: In the decision stage, the similarity score obtained in the matching stage is compared with a threshold. If the score is greater than the set threshold, the decision is to accept the user, otherwise the user is rejected.

Applications

The most relevant applications of off-line signature biometrics is document authentication, credit card transactions, cheque verification, attendance recording, and forensics. Some of these applications are also shared by on-line signature verification, since in the recent past electronic pads for signature acquisition have been introduced for credit card transactions and e-banking.

On-line signatures can also be employed to generate or to secure cryptographic keys which can be used, for example, to electronically sign a document. Specifically, a key generation scheme has been proposed in [11], while biometric cryptosystems, inspired by the fuzzy vault [12] and the fuzzy commitment [13] cryptographic protocols, have been proposed in [14] and in [15, 16] respectively, with application to signature biometrics.

Experimental Results

In the last few years, there have been several attempts to objectively assess the performance of a signature-based verification system. Some public competitions have been launched such as SVC2004: First International Signature Verification Competition [17], the ICDAR2009 Signature Verification Competition [18], and the BioSecure Signature Evaluation Campaign 2009 (BSEC'2009) [19]. In the

SVC04 competition, two scenarios have been considered: in the first one, only signature coordinates versus time have been considered and in the second one, also additional information including pen orientation and pressure have also been taken into account. The best obtained equal error rates (EERs) were 2.84% and 2.89% for the first and second scenario respectively. EERs equal to 2.85% for on-line data and 9.15% for off-line data were the best results achieved during ICDAR2009 where pen coordinates, pen pressure, and pen inclination angles have been considered as features. For the BSEC'2009 competition, two databases have been employed: one acquired by means of a digitizing tablet, and the second one using a PDA which can be employed in a mobile scenario. More specifically, the dependence of the results from the use of different feature combinations has been analyzed in three different cases: use of the only pen coordinates; of the pen coordinates and pressure; and finally of the pen coordinates, pressure, and pen inclination angles. The use of the only pen coordinates gave an EER equal to 4.93% when using a PDA. When the acquisition is made using a tablet, the use of pen coordinates and pressure granted an EER equal to 1.71%, whereas the use of pen coordinates, pressure, and the pen inclination angles led to a best EER of 2.19%. Moreover, the performance variability has also been studied when using, in the enrollment and in the recognition stage, two different sets of signatures acquired in two different sessions spaced of 4 weeks. It has been shown that time variability decreases the performance irrespective to the features employed.

Recommended Reading

1. Plamondon R, Lorette G (1989) Automatic signature verification and writer identification: The state of the art. Pattern Recognit 22(2):107–131
2. Leclerc F, Plamondon R (1994) Automatic signature verification: The state of the art 1989–1993. Int J Pattern Recognit Artif Intell 8(3):643–660
3. Plamondon R, Srihari SN (2000) Online and off-line handwriting recognition: a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 22(1):63–84
4. Dimauro G, Impedovo S, Lucchese MG, Modugno R, Pirlo G (2004) Recent advancements in automatic signature verification. In: Ninth international workshop on frontiers in handwriting recognition, Kokubunji, Tokyo, IEEE press, New York, pp 179–184
5. Lei H, Govindaraju V (2005) A comparative study on the consistency of features in on-line signature verification. Pattern Recognit Lett 15:2483–2489
6. Yang L, Widjaja BW, Prasad R (1995) Application of hidden Markov models for signature verification. Pattern Recognit 28(2):161–170
7. Fierrez J, Ortega-Garcia J, Ramos D, Gonzalez-Rodriguez J (2007) HMM-based on-line signature verification: feature extraction and signature modeling. Pattern Recognit Lett 28(16):2325–2334
8. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49
9. Kholmatov A, Yanikoglu B (2005) Identity authentication using improved online signature verification method. Pattern Recognit Lett 26(15):2400–2408
10. Faundez-Zanuy M (2007) On-line signature recognition based on VQ-DTW. Pattern Recognit 40:981–992
11. Vielhauer C, Steinmetz R (2004) Handwriting: feature correlation analysis for biometric hashes (Special issue on biometric signal processing 2004). EURASIP J Appl Signal Process 4:542–558
12. Juels A, Sudan M (2006) A fuzzy vault scheme. Design Codes Cryptogr 38(2):237–257
13. Juels A, Wattenberg M (1999) A fuzzy commitment scheme. In 6th ACM conference computer and communication security, Singapore, ACM, New York, pp 28–36
14. Freire-Santos M, Fierrez-Aguilar J, Ortega-Garcia J (2006) Cryptographic key generation using handwritten signature. In: SPIE, defense and security symposium, biometric technologies for human identification, Orlando, vol 6202, pp 225–231
15. Maiorana E, Campisi P, Neri A (2008) User adaptive fuzzy commitment for signature templates protection and renewability. SPIE Journal of Electronic Imaging, JEI, Section on Biometrics: Advances in Security, Usability and Interoperability 17(1): 011011-1–011011-12
16. Maiorana E, Campisi P (2010) Fuzzy commitment for function based signature template protection. IEEE Signal Process Lett 17(3):249–252
17. Yeung DY, Chang H, Xiong Y, George S, Kashi R, Matsumoto T, Rigoll G (2004) SVC2004: first international signature verification competition. In: Proceedings of the ICBA, LNCS/3072, Springer, pp 16–22
18. Blankers VL, Heuvel C, Franke KY, Vuurpijl LG (2009) ICDAR 2009 signature verification competition. In: 10th International conference document analysis and recognition, (ICDAR '09), Barcelona, Spain, pp 1403–1407
19. Houmani N, Garcia-Salicetti S, Mayoue A, Dorizzi B (2009) BioSecure signature evaluation campaign 2009 (BSEC'2009): results. http://biometrics.it-sudparis.eu / BSEC2009 / downloads/BSEC2009_results.pdf (on line access August 2010)

Signcryption

Yevgeniy Dodis
Department of Computer Science, New York University, New York, USA

Synonyms

Public-key authenticated encryption

Related Concepts

►Authenticated Encryption

Definition

Signcryption is a method allowing two parties, each possessing a public/secret-key pair, to securely communicate with each other, while simultaneously ensuring both privacy and authenticity. Therefore, signcryption simultaneously provides the benefits of traditional public-key signature and encryption schemes.

Background

Encryption and signature schemes are fundamental cryptographic tools for providing privacy and authenticity, respectively, in the public-key setting. Traditionally, these two important building blocks of public-key cryptography have been considered as *distinct* entities that may be *composed* in various ways to ensure *simultaneous* message privacy and authentication. However, in the last few years, a new, separate primitive – called *signcryption* [14] – has emerged to model a process simultaneously achieving privacy and authenticity. This emergence was caused by many related reasons. The obvious one is the fact that given that *both* privacy and authenticity are simultaneously needed in so many applications, it makes a lot of sense to invest special effort into designing a tailored, more efficient solution than a mere composition of signature and encryption. Another reason is that viewing authenticated encryption as a separate *primitive* may conceptually simplify the design of complex protocols, which require both privacy and authenticity, as signcryption could now be viewed as an "indivisible" atomic operation. Perhaps most importantly, it was noticed by [2, 3] (following some previous work in the symmetric-key setting [4, 10]) that proper modeling of signcryption is not so obvious. For example, a straightforward composition of signature and encryption might not always work; at least, unless some special care is applied [2]. The main reason for such difficulties is the fact that signcryption is a complex *multi-user* primitive, which opens a possibility for some subtle attacks (discussed below), not present in the settings of stand-alone signature and encryption.

Theory

Defining Signcryption

Syntactically, a signcryption scheme consists of the three efficient algorithms $(\mathsf{Gen}, \mathsf{SC}, \mathsf{DSC})$. The key generation algorithm $\mathsf{Gen}(1^\lambda)$ generates the key-pair $(\mathsf{SDK}_U, \mathsf{VEK}_U)$ for user U, where λ is the security parameter, SDK_U is the signing/decryption key that is kept private, and VEK_U is the verification/encryption key that is made public. The randomized signcryption algorithm SC for user U implicitly takes as input the user's secret key SDK_U, and explicitly takes as input the message m and the identity of the recipient ID_R, in order to compute and output the *signcryptext* on Π. For simplicity, one can consider this identity ID_R to be a public key VEK_R of the recipient R, although ID's could generally include more convoluted information (as long as users can easily obtain VEK from ID). Thus, one writes $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{ID}_R)$ as $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{VEK}_R)$, or simply $\mathsf{SC}_U(m, \mathsf{VEK}_R)$. Similarly, user U's deterministic de-signcryption algorithm DSC implicitly takes the user's private SDK_U, and explicitly takes as input the signcryptext $\tilde{\Pi}$ and the senders' identity ID_S. Again, one can assume $\mathsf{ID}_S = \mathsf{VEK}_S$, and write $\mathsf{DSC}_{\mathsf{SDK}_U}(\Pi, \mathsf{VEK}_S)$, or simply $\mathsf{DSC}_U(\Pi, \mathsf{VEK}_S)$. The algorithm outputs some message $\tilde{m}$, or $\perp$ if the signcryption does not verify or decrypt successfully. Correctness property ensures that for any users S, R, and message m, one has $\mathsf{DSC}_R(\mathsf{SC}_S(m, \mathsf{VEK}_R), \mathsf{VEK}_S) = m$.

It is often useful to add another optional parameter to both SC and DSC algorithms: a *label* L (also termed *associated data* [11]). This label can be viewed as a public identifier that is "inseparably bound" to the message m inside the signcryptext. Intuitively, de-signcrypting the signcryptext Π of m with the wrong label should be impossible, as well as changing Π into a valid signcryptext $\tilde{\Pi}$ of the same m under a different label.

Security of Signcryption

Security of signcryption consists of two distinct components: one ensuring privacy, and the other – authenticity. On a high level, privacy is defined somewhat analogously to the privacy of an ordinary encryption, while authenticity – to that of an ordinary digital signature. Formally, privacy is usually defined as *indistinguishability* of signcryptexts under *chosen ciphertext attack*, while authenticity is usually defined as *existential unforgeability* of signcryptexts under *chosen message attack*.

However, several new issues come up due to the fact that signcryption/de-signcryption take as an extra argument the identity of the sender/recipient. Some of those issues (see [2] for in-depth technical discussion, as well as formal definitions of signcryption) are discussed below:

- *Simultaneous Attacks.* Since the user U utilizes its secret key SDK_U to both send and receive the data, it is reasonable to allow the adversary $\mathcal{A}$ oracle access to both the signcryption and the de-signcryption oracle for user U, irrespective of whether $\mathcal{A}$ is attacking privacy or authenticity of U.
- *Two- vs. Multi-user Setting.* In the simplistic two-user setting, where there are only two users S and R in the network, the explicit identities become redundant. This considerably simplifies the design of secure

signcryption schemes (see below), while providing a very useful intermediate step toward general, multi-user constructions (which are often obtained by adding a simple twist to the basic two-user construction). Intuitively, the security in the two-user model already ensures that there are no weaknesses in the way the message is encapsulated inside the signcryptext, but does not ensure that the message is bound to the identities of the sender and/or recipient. In particular, it might still allow the adversary a large class of *identity fraud* attacks, where the adversary can "mess up" correct user identities without affecting the hidden message.

- *Public Non-Repudiation?* In a regular signature scheme, anybody can verify the validity of the signature, and unforgeability of the signature ensures that a signer S indeed certified the message. Thus, one says that a signcryption scheme provides *non-repudiation* if the recipient can extract a regular (publicly verifiable) digital signature from the corresponding signcryptext. In general, however, it is a priori only clear that the *recipient* R is sure that S sent the message. Indeed, without R's secret key SDK_R others might not be able to verify the authenticity of the message, and it might not be possible for R to extract a regular signature of m. Thus, signcryption does not necessarily provide non-repudiation. In fact, for some applications one might explicitly want *not* to have non-repudiation. For example, S might be willing to send some confidential information to R only under the condition that R cannot convince others of this fact. To summarize, non-repudiation is an optional feature that some schemes support, others do not, and others explicitly avoid!
- *Insider vs. Outsider Security.* In fact, even with R's secret key SDK_R it might be unclear to an observer whether S indeed sent the message m to R, as opposed to R "making it up" with the help of SDK_R. This forms the main basis for distinction between *insider-* and *outsider-secure* signcryption. Intuitively, in an outsider-secure scheme the adversary must compromise communication between two honest users (whose keys he does not know). Insider-secure signcryption protects a given user U even if his partner might be malicious. For example, without U's key, one cannot forge signcryptext from U to any other user R, even with R's secret key. Similarly, if honest S sent $\Pi = \mathsf{SC}_S(m, \mathsf{VEK}_U)$ to U and later exposed his key SDK_S to the adversary, the latter still cannot decrypt Π. Clearly, insider-security is stronger than outsider-security, but might not be needed in a given application. In fact, for applications supporting message repudiation, one typically does not want to have insider-security.

Supporting Long Inputs

Sometimes, it is easier to design natural signcryption schemes supporting short inputs. To make such scheme useful for arbitrary lengths inputs, several methods exist how to create signcryption SC' supporting arbitrarily long inputs from SC, which only supports fixed-length (and much shorter) inputs. One such method was suggested by [8]. It uses a new primitive called *concealment*. A concealment is a publicly known randomized transformation, which, on input m, outputs a *hider* h and a *binder* b. Together, h and b allow one to recover m, but separately, (1) the hider h reveals "no information" about m, while (2) the binder b can be "meaningfully opened" by at most one hider h. Further, one requires $|b| \ll |m|$ (otherwise, one could trivially set $b = m$, $h = \emptyset$). One can then let $\mathsf{SC}'(m) = \langle \mathsf{SC}(b), h \rangle$ (and DSC' is similar). It was shown in [8] that the above method yields a secure signcryption SC'. Further, a simple construction of concealment was given: set $h = E_\tau(m)$, $b = \langle \tau, H(h) \rangle$, where E is a symmetric-key one-time secure encryption (with short key τ) and H is a collision-resistant hash function (with short output).

Generic Composition Schemes

The two natural composition paradigms are "encrypt-then-sign" ($\mathcal{E}t\mathcal{S}$) and "sign-then-encrypt" ($\mathcal{S}t\mathcal{E}$). More specifically, assume Enc is a semantically secure encryption against chosen ciphertext attack, and Sig is an existentially unforgeable signature (with message recovery) against chosen message attack. Each user U has a key for for Sig and Enc. Then the "basic" $\mathcal{E}t\mathcal{S}$ from S to R outputs $\mathsf{Sig}_S(\mathsf{Enc}_R(m))$, while $\mathcal{S}t\mathcal{E}$ – $\mathsf{Enc}_R(\mathsf{Sig}_S(m))$. Additionally, [2] introduced a novel generic composition paradigm for *parallel* signcryption. This paradigm uses any secure *commitment scheme*, which on input m, outputs a commitment c and a decommitment d (where c is both hiding and binding). Such commitment schemes have the property that c gives "no information" about the message m, and, yet, it is computationally hard to "open" one c into two different messages $m_1 \neq m_2$, by providing two decommitments d_1 and d_2. Then "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) outputs a pair $\langle \mathsf{Enc}_R(d), \mathsf{Sig}_S(c) \rangle$. Intuitively, the scheme is private as public c reveals no information about m (while d is encrypted), and authentic since c binds one to m. The advantage of the above scheme over the sequential $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ variants is the fact that expensive signature and encryption operations are performed in parallel. In fact, by using *trapdoor commitments* in place or regular

commitments, most computation in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ – including the expensive computation of both public-key signature and encryption – can be done off-line, even before the message m is known!

It was shown by [2] that all three basic composition paradigms yield an insider-secure signcryption in the two-user model. Moreover, $\mathcal{E}t\mathcal{S}$ is outsider-secure even if Enc is secure only against the chosen plaintext attack, and $\mathcal{S}t\mathcal{E}$ is outsider-secure even if Sig is only secure against no message attack. Clearly, all three paradigms are insecure in the multi-user model, since no effort is made to bind the message m to the identities of the sender/recipient. For example, intercepting a signcryptext of the form $\mathsf{Sig}_S(e)$ from S to R, an adversary $\mathcal{A}$ can produce $\mathsf{Sig}_{\mathcal{A}}(e)$, which is a valid signcryptext from $\mathcal{A}$ to R of the same message m, even though m is *unknown* to $\mathcal{A}$. Jee Hea An et al. [2] suggest a simple solution: When encrypting, always append the identity of the sender to the message, and when signing – of the recipient. For example, a multi-user secure variant of $\mathcal{E}t\mathcal{S}$ is $\mathsf{Sig}_S(\mathsf{Enc}_R(m, \mathsf{VEK}_S), \mathsf{VEK}_R)$. Notice, if Enc and/or Sig support labels, these identities can be part of the label rather than the message.

$\mathcal{S}t\mathcal{E}$ and $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ methods always support non-repudiation, while $\mathcal{S}t\mathcal{E}$ might or might not.

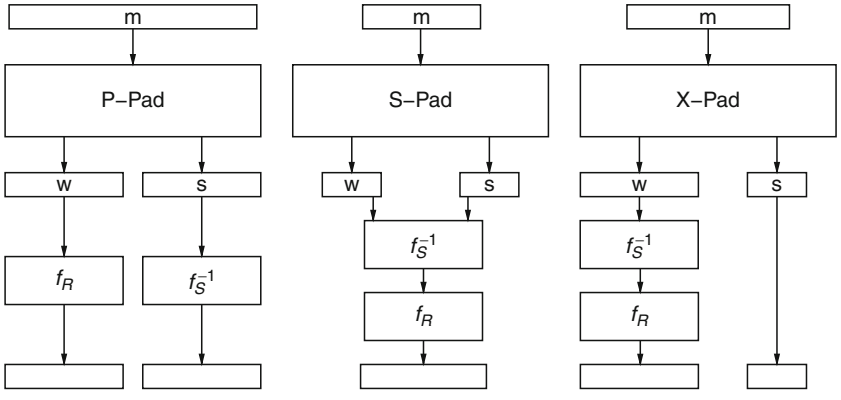
Schemes from Trapdoor Permutations

The generic schemes above validate the fact that signcryption can be built from ordinary signature and encryption, but will be inefficient unless the latter are efficiently implemented. In practice, efficient signature and encryption schemes, such as OAEP [5], OAEP+ [13], PSS-R [6], are built from *trapdoor permutations*, such as RSA, and are analyzed in the random oracle model. Even with these efficient implementations, however, the generic schemes will have several drawbacks. For example, users have to store two independent keys, the message bandwidth is suboptimal and the "exact security" of the scheme is not as good as one might expect. Thus, given that practical schemes are anyway built from trapdoor permutations, it is natural to have highly optimized *direct* signcryption constructions from trapdoor permutations (in the random oracle model).

This is the approach of [9]. In their model, each user U independently picks a trapdoor permutation f_U (together with its trapdoor, denoted f_U^{-1}) and publishes f_U as its public key. (Notice, only a *single* key is chosen, unlike what is needed for the generic schemes.) Then, [9] considers the following three paradigms termed **P**-Pad, **S**-Pad, and **X**-Pad. Each paradigm proceeds by constructing an appropriate *padding scheme* π, applies the padding scheme to the message m, producing two strings $\pi(m) = w\|s$, and then applies the corresponding permutations of the sender and the recipient to w and s, as shown in Fig. 1. Table 1 also shows how the corresponding approaches could be used for plain signature and encryption as well.

The convenience of each padding scheme depends on the application for which it is used. As was shown in [9], **P**-Pad signcryption provides parallel application of "signing" f_S^{-1} and "encrypting" f_R, which can result in efficiency improvements on parallel machines. However, the minimum ciphertext length is twice as large as compared to **S**-Pad, yet the exact security offered by **S**-Pad is not as tight as that of **P**-Pad. Finally, **X**-Pad regains the optimal exact security of **P**-Pad, while maintaining ciphertext length nearly equal to the length of the trapdoor permutation (by achieving quite short s).

It remains to describe secure padding schemes π for **P**-Pad, **S**-Pad, and **X**-Pad. All constructions offered by [9] are quite similar. One starts with any *extractable commitment* (c, d), where c is the commitment and d is the decommitment. Such schemes are very easy to construct in the *random oracle model*. For example, if $|m| = n$, for



Signcryption. Fig. 1 Generalized paddings as used by signcryption

Signcryption. Table 1 Signcryption schemes based on trapdoor permutations

Padding Type	Encryption	Signature	Signcryption
Parallel	$f_R(w)\|s$	$w\|f_S^{-1}(s)$	$f_R(w)\|f_S^{-1}(s)$
Sequential	$f_R(w\|s)$	$f_S^{-1}(w\|s)$	$f_R\left(f_S^{-1}(w\|s)\right)$
e**X**tended sequential	$f_R(w)\|s$	$f_S^{-1}(w)\|s$	$f_R\left(f_S^{-1}(w)\right)\|s$

any $0 \le a \le n$, the following scheme is an extractable commitment: split $m = m_1\|m_2$, where $|m_1| = a$, $|m_2| = n - a$, and set

$$c = G(r) \oplus m_1\|H(m_2\|r)$$
$$d = m_2\|r$$

where G and H are random oracles (with appropriate input/output lengths) and r is a random salt.

To get a secure padding scheme for the **P**-Pad paradigm, one should then apply the Feistel Transform to the resulting pair (d, c), with yet another random oracle F as the round function. Namely, set $w = c$, $s = F(c) \oplus d$. For example, using the extractable commitment above with $a = n$, one gets nothing else but the OAEP padding, while $a = 0$ would give the PSS-R padding! For arbitrary a, [9] call the resulting hybrid between PSS-R and OAEP *Probabilistic Signature-Encryption Padding* (PSEP).

To get the padding π sufficient for either **S**-Pad or **X**-Pad, one only needs to perform one more Feistel round to the construction above: $w' = s$, $s' = F'(s) \oplus w$, and set $\pi(m) = w'\|s'$. Coincidentally, the resulting π also gives a very general construction of the *universal padding schemes* [7].

As described, the Feistel-based padding schemes above would only give the insider security in the two-user setting. To get multi-user security, all one needs to do is to prepend the pair $(\mathsf{VEK}_S, \mathsf{VEK}_R)$ to all the inputs to the random oracles F and F': namely, create effectively independent F and F' for every sender-recipient pairing! More generally, the paddings above also provide label support, if one sticks the label L as part of the inputs to F and F'.

Notice, **P**-Pad, **S**-Pad, and **X**-Pad always support non-repudiation.

Schemes based on Gap Diffie–Hellman

Another popular method to design efficient signcryption scheme is based on the so called *Gap Diffie–Hellman* assumption. Given a cyclic group G of prime order q, and a generator g of G, the assumption states that the *computational Diffie–Hellman* problem (CDH) is computationally hard, even if one is given oracle access to the *decisional Diffie–Hellman* (DDH) oracle. Specifically, it is hard to compute g^{ab} from g^a and g^b, even if one can test whether a tuple $\langle g^x, g^y, g^z\rangle$ satisfies $z = xy \bmod q$.

In both schemes, the user U chooses a random $x_U \in \mathbb{Z}_q$ as its secret key VEK_U, and sets its public key $\mathsf{SDK}_U = y_U = g^{x_U}$. The scheme of [1] is based on the following noninteractive key agreement between users S and R. Namely, both S and R can compute the quantity $Q_{SR} = g^{x_R x_S} = y_S^{x_R} = y_R^{x_S}$. They then set the key $K_{SR} = H(Q_{SR})$, where H is a random oracle, and then always use K_{SR} to perform symmetric-key authenticated encryption of the message m. For the latter, they can use any secure symmetric-key scheme, like "encrypt-then-mac" [4] or OCB [12]. The resulting signcryption scheme can be shown to be outsider-secure for both privacy and authenticity, in the multi-user setting. Clearly, it is not insider-secure, since both S and R know the key K_{SR}. In fact, the scheme is perfectly repudiable, since all the signcryptexts from S could have been easily faked by R.

To get insider-security for authenticity under the same assumption, one can instead consider the following scheme, originally due to [14], but formally analyzed by [3]. Below G and H are random oracles with appropriate domains, and E is a one-time secure symmetric-key encryption (e.g., one-time pad will do). To signcrypt a message from S to R, S chooses a random $x \in \mathbb{Z}_q$, computes $Q = y_R^x$, makes a symmetric key $K = H(Q)$, sets $c \leftarrow E_K(m)$, computes the "validation tag" $r = G(m, y_A, y_B, Q)$ and finally $t = x(r + x_S)^{-1} \bmod q$. Then S outputs $\langle c, r, t\rangle$ as the signcryption of m. To de-signcrypt $\langle c, r, t\rangle$, R first recovers g^x via $w = (y_S g^r)^t$, then recovers the Diffie–Hellman key $Q = w^{x_R}$, the encryption key $K = H(Q)$, and the message $m = D_K(c)$. Before outputting m, however, it double checks if $r = G(m, y_A, y_B, Q)$. While this scheme is insider-secure for authenticity, it is still not insider-secure for privacy.

The scheme above supports public non-repudiation. All that R has to do is to reveal Q, m, and a proof that $Q = w^{x_R}$ (which can be done noninteractively using the *Fiat–Shamir heuristics*, applied to the three-move proof that $\langle g, y_R, w, Q\rangle$ form a DDH-tuple).

Applications

Signcryption is a general way to ensure private and authentic communication between any pair of parties sharing a public-key infrastructure. As such, it is useful in any application where such communication is desired. Additionally, the use of signcryption as a primitive may conceptually simplify the design of complex protocols that require both

privacy and authenticity, as signcryption could now be viewed as an "indivisible" atomic operation.

Recommended Reading

1. An JH (2001) Authenticated encryption in the public-key setting: security notions and analyses. Report 2001/079, Cryptology ePrint Archive
2. An JH, Dodis Y, Rabin T (2002) On the security of joint signature and encryption. In: Knudsen L (ed) Advances in cryptology—EUROCRYPT 2002, Lecture notes in computer science. Springer-Verlag, Berlin. Available from http://eprint.iacr.org/2002/046/
3. Baek J, Steinfeld R, Zheng Y (2002) Formal proofs for the security of signcryption. In: Naccache D, Pailler P (ed), In: 5th International workshop on practice and theory in public key cryptosystems – PKC 2002. Lecture notes in computer science, vol 2274. Springer-Verlag, Berlin
4. Bellare M, Namprempre C (2000) Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto T (ed) Advances in Cryptology – ASIACRYPT 2000, Kyoto, Japan. Lecture notes in computer science, vol 1976. Springer-Verlag, Berlin, pp 531–545
5. Bellare M, Rogaway P (1995) Optimal asymmetric encryption. In: De Santis A (ed) Advances in cryptology – EUROCRYPT 94. Lecture notes in computer science, vol 950. Springer-Verlag, Berlin, pp 92–111. Revised version available from http://www-cse.ucsd.edu/users/mihir/
6. Bellare M, Rogaway P (1996) The exact security of digital signatures: how to sign with RSA and Rabin. In: Maurer U (ed) Advances in cryptology—EUROCRYPT 96. Lecture notes in computer science, vol 1070. Springer-Verlag, Berlin, pp 399–416. Revised version appears in http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html.
7. Coron J-S, Joye M, Naccache D, Pailler P (2002) Universal padding schemes for RSA. In: Yung M (ed) Advances in cryptology – CRYPTO 2002. Lecture notes in computer science. Springer-Verlag, Berlin. Available from http://eprint.iacr.org/2002/115/
8. Dodis Y, An JH (2003) Concealment and its applications to authenticated encryption. In: Biham E (ed) Advances in cryptology – EUROCRYPT 2003. Lecture notes in computer science. Springer-Verlag, Berlin
9. Dodis Y, Freedman MJ, Jarecki S, Walfish S (2004) Versatile padding schemes for joint signature and encryption. In: Pfitzmann B (ed) Eleventh ACM conference on computer and communication security. ACM, New York, pp 196–205
10. Krawczyk H (2001) The order of encryption and authentication for protecting communications (or: How secure is ssl?). In: Kilian J (ed) Advances in cryptology – CRYPTO 2001, Lecture notes in computer science, vol 2139. Springer-Verlag, Berlin, pp 310–331
11. Rogaway P (2002) Authenticated-encryption with associated-data. In: Sandhu R (ed) Ninth ACM conference on computer and communication security. ACM, New York
12. Rogaway P, Bellare M, Black J, Krovetz T (2001) OCB: A blockcipher mode of operation for efficient authenticated encryption. In: Samarati P (ed) Eighth ACM conference on computer and communication security. ACM, New York, pp 196–205. Full version available from http://www.cs.ucsdavis.edu/~rogaway
13. Shoup V (2001) OAEP reconsidered. In: Kilian J (ed) Advances in Cryptology – CRYPTO 2001, Lecture notes in computer science, vol 2139. Springer-Verlag, Berlin, pp 240–259
14. Zheng Y (1997) Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). In: Kaliski Jr. BS (ed) Advances in cryptology – CRYPTO '97. Lecture notes in computer science, vol 1294. Springer-Verlag, Berlin, pp 165–179

Signed Digit Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Synonyms

Signed window exponentiation

Related Concepts

►Exponentiation Algorithms

Definition

Signed digit exponentiation computes powers using exponent representations that involve both positive and negative digits. The extended digit range can make exponentiations faster.

Background

Signed digit exponentiation is an approach for computing powers in any ►group in which the inverse A^{-1} of any group element A can be computed quickly (such as the groups of points on an elliptic curve employed in ►elliptic curve cryptography). It is related to ►sliding window exponentiation: while in sliding window exponentiation each window corresponds to a positive digit value, signed digit exponentiation additionally makes use of the corresponding negative digit values, and the ease of inversion makes these extra digits available almost for free. This often makes signed digit exponentation faster when using the same amount of memory for storing group elements, and allows it to reach approximately the same speed with less memory.

Let $B_k = \{\pm 1, \pm 3, \ldots, \pm(2^k - 1)\}$ where k is a positive integer; and let a base-two representation of an exponent e be given using the digit set $\{0\} \cup B_k$, that is,

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0\} \cup B_k.$$

Assuming that l is chosen such that $e_{l-1} \neq 0$, the *left-to-right signed digit exponentiation method* computes g^e as follows where g is any group element; cf. the left-to-right ►sliding window exponentiation method.

$G_1 \leftarrow g$
$A \leftarrow g \circ g$
for $d = 3$ to $2^k - 1$ step 2 **do**
  $G_d \leftarrow G_{d-2} \circ A$

if $e_{l-1} > 0$ **then**
  $A \leftarrow G_{e_{l-1}}$
else
  $A \leftarrow G^{-1}_{-e_{l-1}}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i \neq 0$ **then**
    if $e_i > 0$ **then**
      $A \leftarrow A \circ G_{e_i}$
    else
      $A \leftarrow A \circ G^{-1}_{-e_i}$
return A

The *right-to-left signed digit exponentiation method* computes g^e as follows; cf. the right-to-left ►sliding window exponentiation method. Note that the algorithm as written can be optimized similarly to the right-to-left ►2^k-ary exponentiation or ►sliding window exponentiation methods to avoid (at least) 2^{k-1} applications of the group operation.

for $d = 1$ to $2^k - 1$ step 2 **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$

for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    if $e_i > 0$ **then**
      $B_{e_i} \leftarrow B_{e_i} \circ A$
    else
      $B_{-e_i} \leftarrow B_{-e_i} \circ A^{-1}$
  if $i < l - 1$ **then**
    $A \leftarrow A \circ A$

{Now $g^e = \prod_{d \in \{1,3,\ldots,2^k-1\}} B_d^d$.}

for $d = 2^k - 1$ to 3 step -2 **do**
  $B_{d-2} \leftarrow B_{d-2} \circ B_d$
  $B_1 \leftarrow B_1 \circ (B_d \circ B_d)$
return B_1

For both the left-to-right and the right-to-left variant, it remains to be considered how signed digit representations of exponents e using the digit set $\{0\} \cup B_k$ with $B_k = \{\pm 1, \pm 3, \ldots, \pm 2^k - 1\}$ can be obtained. An algorithm for the simplest case $k = 1$ is due to Reitwiesner [1]; the representation obtained by it (using digits $\{-1, 0, 1\}$) is known as the *non-adjacent form* (*NAF*) of e. The generalization for an arbitrary parameter k was simultaneously suggested by multiple researchers; the following algorithm is from [2]:

$c \leftarrow e$
$i \leftarrow 0$
while $c > 0$ **do**
  if c is odd **then**
    $d \leftarrow c \bmod 2^{k+1}$
    if $d > 2^k$ **then**
      $d \leftarrow d - 2^{k+1}$
    $c \leftarrow c - d$
  else
    $d \leftarrow 0$
  $e_i \leftarrow d$; $i \leftarrow i + 1$
  $c \leftarrow c/2$
return $e_{i-1}, \ldots, e_0$

This algorithm is a variant of right-to-left scanning with an effective window size of $k+1$. For efficiency considerations, if the cost of inverting groups elements and the additional cost for obtaining the appropriate representation of e can be neglected, signed digit exponentiation differs from ►sliding window exponentiation with the same parameter k in that the expected number of nonzero digits in the representation is approximately $l/(k+2)$ instead of approximately $l/(k+1)$ (but the maximum possible length of the signed digit representation is longer: while l cannot exceed the length of the binary representation of e for sliding window exponentiation, it can be said length plus 1 for signed digit exponentiation).

Applications

Signed digit exponentiation is useful in ►elliptic curve cryptography. When ►side-channel attacks have to be considered to protect secret values, specific variants of the basic methods may be needed.

Recommended Reading

1. Reitwiesner GW (1960) Binary arithmetic. Adv Comput 1: 231–308
2. Solinas JA (2000) Efficient arithmetic on koblitz curves. Design Code Cryptogr 19:195–249

Signed Window Exponentiation

►Signed Digit Exponentiation

SIM/UICC

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Subscriber identity module; UMTS IC card; Universal integrated circuit card

Definition

A subscriber identification module (SIM) on a removable SIM card stores securely the international mobile subscriber identity (IMSI) used to identify a subscriber on mobile devices (such as mobile phones and computers) in a GSM (Global System for Mobile Communications) network.

The universal integrated circuit card (UICC) is the ►*chip card*, a multi-application platform, used in mobile devices in GSM and UMTS (Universal Mobile Telecommunications System, one of the third-generation (3G) mobile telecommunications technologies) networks. In a GSM network, the UICC contains a SIM application and in a UMTS network it is the USIM application. A UICC may contain several applications, making it possible for the same chip card to give access to both GSM and UMTS networks, but also to store a variety of other applications.

Background

Since its conception in 1988 the SIM has undergone continuous development extending its technical and functional capabilities. Initially it was defined as a security module to authenticate the user to the network providing, at the same time, some very limited amount of memory for network and private user data. Smart cards were in those days still in their infancy. The technological and market requirements of GSM shaped the smart card SIM, which offers more than just security. They are a secure platform for mobile network operator defined multiple services. The standardization of the SIM and the GSM was done in ETSI (see Ref. [2]).

In 1999, the standardization work started on the specification of the next-generation SIM, the USIM, a true multi-application module, involving next to ETSI, worldwide major mobile communication systems standards bodies within 3GPP. At the same time, a split was made of the core specifications into an application-independent part and an application-specific part. The UICC refers to the multi-application platform, the so-called carrier (the application-independent part).

GSM : SIM = physical card + GSM application (GSM11.11)

3G : UICC = physical card and basic logical functionality

and USIM = 3G application on a UICC

See Ref. [1] for more information.

Theory

A SIM is the physically secured module, which contains the IMSI, an authentication algorithm, the authentication key, and other (security related) information and functions. The basic function of the SIM is to authenticate the subscriber identity in order to prevent misuse of the mobile station (MS) and the network.

The UICC is a multi-application platform smart card consisting of generic (application independent) functions and features with a clear separation of lower layers and applications. The UICC security architecture is designed so as to be able to provide, if necessary, a multi-verification environment, that is, an environment in which the card can have more than one first-level application and may support separate user verification requirements for each application.

A universal subscriber identity module (USIM) is an application for UMTS mobile telephony running on a UICC smart card which is inserted in a 3G mobile phone. There is a common misconception to call the UICC itself a USIM, but the USIM is merely a logical entity on the physical card. Similar to the SIM, it stores user subscriber information (IMSI), authentication information and provides storage space for text messages and phone book contacts.

SIM/UICC cards are available in three standard sizes. The first is the size of a credit card (ISO/IEC 7810 ID-1). Soon the race for smaller mobile phones called for a smaller version of the card. The newer, most popular miniature version has the same thickness, but has a length of 25 mm and a width of 15 mm (ISO/IEC 7810 ID-000), and has one of its corners truncated (chamfered) to prevent misinsertion. The newest incarnation known as the 3FF or micro-SIM has dimensions of 15 mm × 12 mm. Most cards of the two smaller sizes are supplied as a full-sized card with the smaller card held in place by a few plastic links; it can easily be broken off to be used in a device that uses the smaller SIM.

Since the card slot is standardized, a subscriber can easily move their SIM/UICC from one handset to another. Similarly, usually a subscriber can change carriers by inserting a new carrier's SIM/UICC card into their existing handset.

Applications

The SIM is to be considered as a mono-application platform supporting value-added services as part of the GSM application offered by the mobile network operator (MNO), the issuer of the SIM. The UICC is a true multi-application platform supporting independent applications running in parallel, from different service providers. Although the MNO is typically the issuer of the UICC, the service providers may have full control over their application residing on the UICC. These applications range from banking, transport, identity, loyalty, etc. See Refs. [3, 4] for more information.

The integration of the ETSI framework for UICC and the application management framework of GlobalPlatform is standardized in the UICC configuration (see Ref. [4]).

Recommended Reading

1. Hillebrand F (2002) GSM and UMTS: the creation of global mobile communication. Wiley, Chichester
2. http://www.etsi.org
3. http://www.gsmworld.com/
4. http://www.globalpatform.org
5. http://www.3gpp.org

Simultaneous Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Synonyms

Multi-exponentiation; Shamir's trick

Related Concepts

►Exponentiation Algorithms

Definition

Simultaneous exponentation means computing a power product without computing the individual powers.

Background

Various schemes for public key cryptography involve computing power products in some commutative ►group (or commutative semigroup). A straightforward way to compute a power product

$$\prod_{j=1}^{n} g_j^{e_j}$$

is to compute the individual powers $g_j^{e_j}$ using ►binary exponentiation or some other exponentiation method, and perform $n-1$ applications of the group operation to multiply these partial results. However, specialized algorithms for computing power products are often faster. The task of computing a power product is sometimes called *multi-exponentiation,* and performing a multi-exponentiation by a procedure that does not involve computing the partial results $g_j^{e_j}$ is known as *simultaneous exponentiation.*

Two methods for multi-exponentiation that both generalize left-to-right ►sliding window exponentiation are *simultaneous sliding window exponentiation,* which is due to Yen, Laih, and Lenstra [3] (based on the simultaneous 2^k-ary exponentiation method from Straus [2], colloquially known as *Shamir's trick*), and *interleaved sliding window exponentiation* [1]. Like the sliding window method for single exponentiations, these methods use the binary representation of exponents, on which nonoverlapping windows are placed such that every nonzero bit is covered by one of the windows. Simultaneous sliding window exponentiation and interleaved sliding window exponentiation use different approaches for placing windows; sometimes the former is faster, sometimes the latter.

Theory

Simultaneous sliding window exponentiation uses windows up to some maximum width k that span across all n exponents; for example, for exponents e_1, e_2, e_3 with binary representations 1011010, 0011001, and 1001011 and $k = 2$:

e_1	1	0	1 1	0	1 0	
e_2	0	0	1 1	0	0 1	
e_3	1	0	0 1	0	1 1	

Such windows can be found by left-to-right scanning: look at the binary representations of the exponents simultaneously, going from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that one of the rightmost bits is also nonzero. The result of collapsing the window values into the right-most row of each window can be considered a base-two representation

$$(e_1, \ldots, e_n) = \sum_{i=0}^{l-1} (e_{1,i}, \ldots, e_{n,i}) 2^i$$

of the vector of exponents, such as

$$\begin{array}{ll} e_1 & 1\ 0\ 0\ 3\ 0\ 0\ 2 \\ e_2 & 0\ 0\ 0\ 3\ 0\ 0\ 1 \\ e_3 & 1\ 0\ 0\ 1\ 0\ 0\ 3 \end{array}$$

for the above example. Assume such a representation is given with l chosen minimal, that is, $(e_{1,l}, \ldots, e_{n,l}) \neq (0, \ldots, 0)$. To perform a simultaneous sliding window exponentiation, first products

$$G_{(d_1,\ldots,d_n)} = \prod_{j=1}^{n} g_j^{d_j}$$

of small powers are computed and stored for all possible window values, namely, for the tuples $(d_1, \ldots, d_n)$ with $d_j \in \{0, 1, \ldots, 2^k - 1\}$ for $j = 1, \ldots, n$ such that at least one of the d_j is odd. There are $2^{nk} - 2^{n(k-1)}$ such tuples, and computing the table of those products can be done with

$$2^{nk} - 2^{n(k-1)}$$

applications of the group operation, n of which are squarings ($g_1, \ldots, g_n$ appear in the table and are available without any computation; once $g_j \circ g_j$ for $j = 1, \ldots, n$ have been computed as temporary values, each of the $2^{nk} - 2^{n(k-1)} - n$ remaining table values can be obtained by using the group operation once). The multi-exponentiation result then is computed using the table of small powers:

$A \leftarrow G_{(e_{1,l-1},\ldots,e_{n,l-1})}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $(e_{1,l}, \ldots, e_{n,l}) \neq (0, \ldots, 0)$ **then**
    $A \leftarrow A \circ G_{(e_{1,i},\ldots,e_{n,i})}$
return A

For random b-bit exponents, this requires at most another $b - 1$ squaring operations and on average approximately another

$$b \cdot \frac{1}{k + \frac{1}{2^n - 1}}$$

general group operations. Note that in practice it is not necessary to completely derive the representation

$$(e_{1,l-1}, \ldots, e_{n,l-1}), \ldots, (e_{1,0}, \ldots, e_{n,0})$$

before starting the exponentiation; instead, left-to-right scanning can be used to determine it window by window when it is needed.

In *interleaved sliding window exponentiation,* each single exponent has independent windows up to some maximum width k; for example, for exponents e_1, e_2, e_3 with binary representations 1011010, 0011001, and 1001011 and $k = 3$:

$$\begin{array}{ll} e_1 & \boxed{1\ 0\ 1}\boxed{1\ 0\ 1}\ 0 \\ e_2 & 0\ 0\ \boxed{1\ 1}\ 0\ 0\ \boxed{1} \\ e_3 & \boxed{1}\ 0\ 0\ \boxed{1\ 0\ 1}\boxed{1} \end{array}$$

For each exponent, such windows can be found by left-to-right scanning: look at the binary representation of the respective exponent, going from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that the rightmost bit is also non-zero. To perform an interleaved sliding window exponentiation, first for each g_j, the powers for odd exponents 1 up to $2^k - 1$ are computed and stored:

for $j = 1$ to n **do**
  $G_{j,1} \leftarrow g$
  $A \leftarrow g \circ g$
  for $d = 3$ to $2^k - 1$ step 2 **do**
    $G_{j,d} \leftarrow G_{j,d-2} \circ A$

Then the multi-exponentiation result is computed using that table of powers. The following algorithm shows how this computation can be implemented including left-to-right scanning of exponents up to b bits. The algorithm accesses the bits $e_j[i]$ of the binary representations

$$e_j = \sum_{i=0}^{b-1} e_j[i] 2^i, \quad e_j[i] \in \{0, 1\}$$

of the exponents; the notation $e_j[i \ldots h]$ is shorthand for $\sum_{v=h}^{i} e_j[v] 2^{v-h}$.

$A \leftarrow$ identity element
for $j = 1$ to n **do**
  $window\_position_j \leftarrow -1$
for $i = b - 1$ down to 0 **do**
  $A \leftarrow A \circ A$
  for $j = 1$ to n **do**
    if $window\_position_j = -1$ and $e_j[i] = 1$ **then**
      $h \leftarrow i - k + 1$
      if $h < 0$ **then**
        $h \leftarrow 0$
      while $e_j[h] = 0$ **do**
        $h \leftarrow h + 1$
      $window\_position_j \leftarrow h$
      $E_j \leftarrow e_j[i \ldots h]$
    if $window\_position_j = i$ **then**
      $A \leftarrow A \circ G_{j,E_j}$
      $window\_position_i \leftarrow -1$
return A

The algorithm as written can be improved by a simple optimization: while A still has its initial value, omit the statement $A \leftarrow A \circ A$, and use a direct assignment $A \leftarrow G_{j,E_j}$ instead of the first assignment $A \leftarrow A \circ G_{j,E_j}$. With this optimization, an interleaved sliding window exponentiation takes up to $n + b - 1$ squarings and on average about

$$n \cdot \left(2^{k-1} - 1 + \frac{b-1}{k+1}\right)$$

general group operations.

Interleaved sliding window exponentiation essentially interleaves the operations of n single exponentiations using left-to-right ►sliding window exponentiation, saving many of the squarings. In groups where computing inverses of elements is possible very quickly, it is possible to similarly interleave the operations of n single exponentiations using left-to-right ►signed digit exponentiation for faster multi-exponentiation.

Applications

Simultaneous exponentiation has various applications in ►public key cryptography, such as for verifying signatures for the DSA scheme from the ►Digital Signature Standard.

Recommended Reading

1. Möller B (2001) Algorithms for multi-exponentiation, selected areas in cryptography – SAC 2001. LNCS, vol 2259. Springer, Berlin, pp 165–180
2. Straus EG (1964) Problems and solutions: addition chains of vectors. Am Math Mon 71:806–808
3. Yen S-M, Laih C-S, Lenstra AK (1994) Multi-exponentiation. IEE Proc – Comput and Digit Tech 141:325–326

Simultaneous Transactions

►Fair Exchange

Single Euro Payments Area

►SEPA

Single Wire Protocol

►SWP

Skipjack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Impossible Differential Attack

Definition

Skipjack [6] is the secret key encryption algorithm (►Symmetric Cryptosystem) developed by the NSA for the Clipper chip initiative (including the Capstone chip and the Fortezza PC card).

Background

It was implemented in tamper-resistant hardware and its structure was kept secret since its introduction in 1993.

On June 24, 1998, Skipjack was declassified, and its description was made public on the Web site of NIST [6].

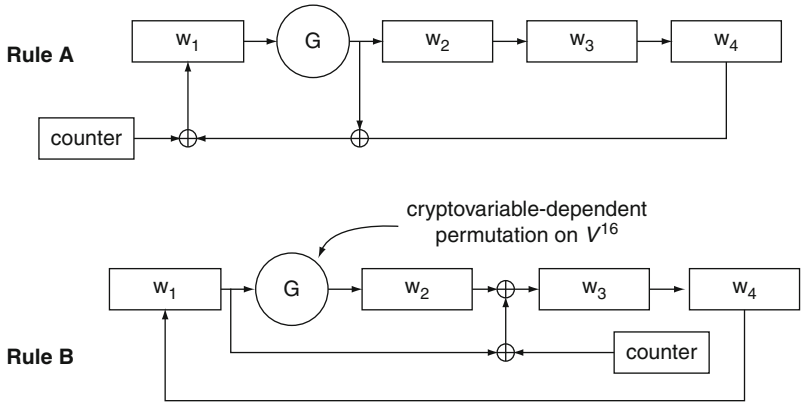
Theory

It is an *iterative* block cipher with 64-bit block, 80-bit key, and 32 rounds. It has two types of rounds, called Rule A and Rule B. Each round is described in the form of a linear feedback shift register with an additional nonlinear keyed G permutation. Rule B is basically the inverse of Rule A with minor positioning differences. Skipjack applies eight rounds of Rule A, followed by eight rounds of Rule B, followed by another eight rounds of Rule A, followed by another eight rounds of Rule B. The original definitions of Rule A and Rule B are given in Fig. 1, where *counter* is the round number (in the range 1–32), G is a four-round Feistel permutation whose F function is defined as an 8×8-bit S box, called *F Table*, and each round of G is keyed by eight bits of the key.

The key schedule of Skipjack takes a 10-byte key, and uses four of them at a time to key each G permutation. The first four bytes are used to key the first G permutation, and each additional G permutation is keyed by the next four bytes cyclically, with a cycle of five rounds.

Application

Skipjack has been subject to intensive analysis [2, 4–6]. For example, Skipjack reduced to (the first) 16 rounds can be attacked with 2^{17} chosen plaintexts and 2^{34} time of analysis [6], which may be reduced to 2^{14} texts and 2^{16} steps using the *yoyo-game* approach [1]. Attacking the



Skipjack. Fig. 1 SKIPJACK stepping rules

middle 16 rounds of Skipjack requires only 3 chosen plaintexts and 2^{30} time of analysis. The currently most successful attack against the cipher is the impossible differential attack which breaks 31 rounds out of 32, marginally faster than exhaustive search [2]. A summary of the state of the art and of the many flawed attacks is given in [8].

In addition, it is worth noting that Skipjack can be attacked by a generic time-memory trade-off approach requiring 2^{80} steps of precomputation and 2^{54} – 80-bit words (i.e., 2^{60} bits) of memory, but then each search for a key requires only 2^{54} steps of computation. In the case of a *multiple-target* attack, if the attacker is given encryptions of an arbitrary fixed known plaintext under 2^{32} different keys, then he can recover one of these keys using *time-memory-data trade-off* attack after 2^{48} steps of precomputation (which is done only once) and in 2^{32} time and memory. Such attack applies to any 80-bit cipher [3].

Recommended Reading

1. Biham E, Biryukov A, Dunkelman O, Richardson E, Shamir A (1999) Initial observations on skipjack: Cryptanalysis of Skipjack-3XOR. In: Tavares SE, Meijer H (eds) Selected areas in cryptography – SAC'98. Lecture notes in computer science, vol 1556. Springer, Berlin, pp 362–376
2. Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern J (ed) Advances in Cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 12–23
3. Biryukov A, Mukhopadhyay S, Sarkar P (2005) Improved Time-Memory Trade-offs with Multiple Data". In: Proceedings of SAC'2005. Lecture notes in computer science
4. Granboulan L (2001) Flaws in differential cryptanalysis of Skipjack. In: Matsui M (ed) Proceedings of fast software encryption – FSE 2001. Lecture notes in computer science, vol 2335. Springer, Berlin, pp 328–335
5. Hwang K, Lee W, Lee S, Lee S, Kim J (2002) Saturation attacks on round Skipjack. In: Daemen J, Rijmen V (eds) Fast software encryption – FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 100–111
6. Knudsen LR, Robshaw M, Wagner D (1999) Truncated differentials and Skipjack. In: Wiener M (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 165–180
7. NIST (1998) SKIPJACK and KEA algorithm specification. Technical Report, http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm.Version2.0
8. Kim J, Phan R (2009) A cryptanalytic view of the NSA's Skipjack Block Cipher Design. In: Advances in information security and assurance (ISA). Lecture notes in computer science, vol 5576. Springer, Berlin, pp 368–381

Slide Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►DES-X

Definition

Slide attack is generic attack designed by Biryukov and Wagner [1, 2]. It can be applied in both known plaintext or chosen plaintext scenarios. It can be viewed as a variant of a related key attack, in which a relation of the key with itself is exploited. The main feature of this attack is that it realizes a dream of cryptanalysts: if the cipher is vulnerable to such an attack, the complexity of the attack is independent of the number of rounds of the cipher. A typical slide of one encryption against another by one round (under the same key) is shown in Fig. 1.

$$P_0 \rightarrow F_1\ F_2\ F_3 \ldots F_r \rightarrow C_1$$
$$P_1 \rightarrow F_1\ F_2\ F_3 \ldots F_r \rightarrow C_2$$

Slide Attack. Fig. 1 A typical slide attack

Theory

If the equation $F_1(P_0, K_1) = P_1$ holds, the pair is called a *slid pair*. The attacker would then obtain two equations:

$$F_1(P_0, K_1) = P_1, \qquad F_r(C_0, K_r) = C_1,$$

where the second equation would hold for free due to sliding. These equations involve only a single round function, and thus could be solved by the attacker for the secret subkeys K_1, K_r of these rounds. The attacker may create properly *slid pairs* (P_0, P_1) by birthday paradox or by careful construction. For an arbitrary cipher, the attack has complexity of $2^{n/2}$ known-plaintexts, where n is the blocksize. For a Feistel cipher, complexity is reduced to $2^{n/4}$ chosen plaintexts.

Several ciphers or slight modifications of existing ciphers have been shown vulnerable to such attacks, for example, the Brown-Seberry variant of the Data Encryption Standard (DES) [3] (rotations in key-schedule are by 7 bits, instead of varying 1, 2 rotations as in the original DES), DES-X, the *Even-Mansour scheme* [4], arbitrary Feistel ciphers with 4-round periodic key-schedule as well as round-reduced versions of GOST. The basic attack has been extended into a *slide-with a twist*, a technique where encryption is slid against decryption and *complementary slide* technique [2], where the inputs to the rounds do not have to be identical but may have the difference which is canceled out by a difference in the keys. In the same paper, another generalization of the technique is given which allows to get many slid pairs simultaneously by cyclic reencryption and thus allows to attack strong round functions. This technique has received an interesting extension in [5], which allows fast detection and generation of many slid pairs if the attacker has access to almost all the codebook of the cipher. Such attack is meaningful if the key size of the cipher is larger than the block size.

Applications

It is clear that slide-attack would apply to any *iterative* construction which has enough *self-similarity* in its rounds. It could be applied to block-ciphers as described above, to stream-ciphers (see, e.g., resynchronization attack on *WAKE-ROFB* [1]) or to MAC and hash functions (see, e.g., a *slid pair* discovery for SHA-1 by Saarinen [6]. Slide attacks have been applied to some more stream ciphers and hash functions recently (see, e.g., [7]).

In practice, the attack seems easy to avoid by breaking the similarity of the round transforms by applying round counters (as is done for example in Skipjack) or different random constants in each round (as in Rijndael/AES, SHA-256, and many other constructions). Whether such simple changes are indeed sufficient is a matter of further research.

Recommended Reading

1. Biryukov A, Wagner D (1999) Slide attacks. In: Knudsen LR (ed) Proceedings of Fast Software Encryption – FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 245–259
2. Biryukov A, Wagner D (2000) Advanced slide attacks. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 589–606
3. Brown L, Seberry J (1990) Key scheduling in DES type cryptosystems. In: Seberry J, Pieprzyk J (eds) Advances in cryptology – AUSCRYPT'90. Lecture notes in computer science, vol 453. Springer, Berlin, pp 221–228
4. Even S, Mansour Y (1997) A construction of a cipher from a single pseudorandom permutation. J Cryptol 10(3):151–162
5. Biham E, Dunkelman O, Keller N (2007) Improved Slide Attacks. In: Biryukov A (ed) Proceedings of fast software encryption – FSE'07. Lecture notes in computer science, vol 4593. Springer, Berlin, pp 153–166
6. Saarinen M-JO (2003) Cryptyanalysis of block ciphers based on SHA-1 and MD5. In: Johansson T (ed) Proceedings of fast software encryption – FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 36–44
7. Gorski M, Lucks S, Peyrin T (2008) Slide attacks on a class of hash functions. In: Preneel B (ed) Advances in cryptology – ASIACRYPT 2008. Lecture notes in computer science, vol 5350. Springer, Berlin, pp 143–160

Sliding Window Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Related Concepts

►Exponentiation Algorithms

Definition

Sliding window exponentiation computes powers by looking at a number of exponent bits at a time: a window onto the binary representation of the exponent. Letting the window slide, as opposed to using fixed window positions, provides for performance improvements.

Background

Sliding window exponentiation is an approach for computing powers in any ►group (or semigroup). Like ►2^k-ary exponentiation, it generalizes ►binary exponentiation and

is parameterized by a positive integer k, where the case $k = 1$ is the same as binary exponentiation. Sliding window exponentiation can be considered an improved variant of 2^k-ary exponentiation: with identical $k \geq 2$, sliding window exponentiation needs storage for fewer group elements and usually performs less applications of the group operation than 2^k-ary exponentiation. However, the algorithms for sliding window exponentiation are slightly more complicated.

Theory

2^k-ary exponentiation uses the 2^k-ary representation of exponents, which can be considered as looking at the binary representation through fixed windows of width k:

$$\boxed{0\ 0\ 1}\boxed{1\ 1\ 0}\boxed{1\ 0\ 0}\boxed{0\ 1\ 1}\boxed{0\ 0\ 1}\boxed{0\ 1\ 0}$$

Sliding window exponentiation improves over this based on the observation that fewer windows of width up to k can suffice to cover all nonzero exponent bits if one allows the windows to take arbitrary positions. Also, one can arrange for all windows to be odd-valued (i.e., have a 1 as the rightmost bit). Then the bits covered by each single window correspond to a value in the set $B_k = \{1, 3, \ldots, 2^k - 1\}$, and the number of possible window values is less than with the 2^k-ary exponentiation method. Covering the binary representation of the exponent by such windows yields a base-two representation of the exponent that uses the digit set $\{0\} \cup B_k$. One possible way to determine windows for a given exponent is to look at the binary representation of the exponent from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that the rightmost bit is also nonzero:

$$\begin{array}{rl} & 0\ 0\ \boxed{1\ 1\ 1}\ 0\ \boxed{1}\ 0\ 0\ 0\ \boxed{1\ 1}\ 0\ 0\ \boxed{1\ 0\ 1}\ 0 \\ \Rightarrow & \qquad\quad 7\ 0\ 1\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 5\ 0 \end{array}$$

Another possibility is to look at the binary representation of the exponent from right to left, starting a new width-k window whenever a nonzero bit is encountered:

$$\begin{array}{rl} & 0\ \boxed{0\ 1\ 1}\boxed{1\ 0\ 1}\ 0\ 0\ \boxed{0\ 1\ 1}\ 0\ 0\ \boxed{1\ 0\ 1}\ 0 \\ \Rightarrow & \qquad 3\ 0\ 0\ 5\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 5\ 0 \end{array}$$

Such left-to-right scanning or right-to-left scanning yields a representation

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0\} \cup B_k.$$

In the following, let such a representation of some positive integer e be given with l chosen minimal; thus, $e_{l-1} \neq 0$.

The *left-to-right sliding window exponentiation method* computes g^e, where g is an element of the group (or semigroup), as follows. First the powers for odd exponents 1 up to $2^k - 1$ are computed and stored:

$G_1 \leftarrow g$
$A \leftarrow g \circ g$
for $d = 3$ to $2^k - 1$ step 2 **do**
  $G_d \leftarrow G_{d-2} \circ A$

Then g^e is computed using the table of powers $G_1 = g$, $G_3 = g^3, \ldots, G_{2^k-1} = g^{2^k-1}$:

$A \leftarrow G_{e_{l-1}}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i \neq 0$ **then**
    $A \leftarrow A \circ G_{e_i}$
return A

Note that in practice it is not necessary to completely derive the representation $e_{l-1}, \ldots, e_0$ before starting the exponentiation; instead, left-to-right scanning can be used to determine it digit by digit when it is needed without storing it completely. Left-to-right sliding window exponentiation is the practical application of algorithmic ideas from [2].

Like binary exponentiation and 2^k-ary exponentiation, sliding window exponentiation has a variant that performs a ►right-to-left exponentiation:

for $d = 1$ to $2^k - 1$ step 2 **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$
for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    $B_{e_i} \leftarrow B_{e_i} \circ A$
  if $i < l - 1$ **then**
    $A \leftarrow A \circ A$

{Now $g^e = \prod_{d \in \{1,3,\ldots,2^k-1\}} B_d^d$; this can be computed as follows:}

for $d = 2^k - 1$ to 3 step -2 **do**
  $B_{d-2} \leftarrow B_{d-2} \circ B_d$
  $B_1 \leftarrow B_1 \circ (B_d \circ B_d)$
return B_1

Again, in practice it is not necessary to completely derive the representation $e_{l-1}, \ldots, e_0$ before starting the exponentiation; here, right-to-left scanning can be used to determine it digit by digit when it is needed. The algorithm as written can be optimized similarly to the right-to-left ►2^k-ary exponentiation method to avoid (at least) 2^{k-1}

applications of the group operation. The idea used to perform sliding window exponentiation in right-to-left fashion is due to Yao [3]; the sub-algorithm shown above for computing $\prod_{d\in\{1,3,\ldots,2^k-1\}} B_d^d$ is due to Knuth [1, answer to exercise 4.6.3-9].

The number of group operations performed during a sliding window exponentiation with maximum window width k depends on the length l of the sliding window representation and on the number of digits in the representation $e_{l-1},\ldots,e_0$ that are nonzero. For any b-bit exponent ($2^{b-1} \le e < 2^b$), the length l is bounded by $b-k < l \le b$. Assume that left-to-right or right-to-left scanning is performed on a sequence of independently and uniformly random bits; then a new window will be started on average every $k+1$ bits. For b-bit exponents, one bit is necessarily nonzero, and both scanning techniques will usually have an unused part in the final window when the end of the exponent is reached. The expected number of nonzero values among $e_{l-1},\ldots,e_0$ for random b-bit exponents lies between $b/(k+1)$ and $1+(b-1)/(k+1)$.

Using the upper bounds to derive estimates for average performance that are on the safe side (i.e., slightly pessimistic) gives b squaring operations (one time $g\circ g$ and $b-1$ times $A\circ A$) and

$$2^{k-1}-1+\frac{b-1}{k+1}$$

general group operations for left-to-right sliding window exponentiation, or

$$2^{k-1}-2+b$$

squaring operations ($b-1$ times $A\circ A$ and $2^{k-1}-1$ times $B_d\circ B_d$) and

$$\underbrace{1+\frac{b-1}{k+1}}_{\text{loop over } i}+\underbrace{2\cdot(2^{k-1}-1)}_{\text{loop over } d}-\underbrace{2^{k-1}}_{\text{optimization}}=2^{k-1}-1+\frac{b-1}{k+1}$$

general group operations for right-to-left sliding window exponentiation with the optimization explained above.

In some groups, such as those employed in ►elliptic curve cryptography, computing inverses of elements is a very fast operation. For such groups, better performance than with ordinary sliding window exponentiation can often be obtained by using ►signed digit exponentiation instead.

Applications

Exponentiation is frequently used in ►public key cryptography. Applications for which the sliding window method is particulary well suited include verification in the ►RSA digital signature scheme.

Recommended Reading

1. Knuth DE (1998) The art of computer programming, vol 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
2. Thurber EG (1973) On addition chains $l(mn)\le l(n)-b$ and lower bounds for $c(r)$, Duke Math J 40:907–913
3. Yao AC-C (1976) On the evaluation of powers. SIAM J Comput 5:100–103

Smart Card

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Chip card; Integrated circuit card

Related Concepts

►Contactless Cards; ►e-ID card; ►Payment Card; ►SIM/UICC; ►Token

Definition

A smart card is a credit card–sized token containing a microprocessor enabling it to process and store information, to support single or multiple applications, and to operate both off-line and on-line.

Background

In 1968, German rocket scientist Helmut Gröttrup and his colleague Jürgen Dethloff invented the automated chip card, receiving a patent applied for in 1968 and granted only in 1982, while working for the German company Giesecke & Devrient. The first mass use of the cards was a Télécarte for payment in French pay phones, starting in 1983.

French inventor Roland Moreno patented the memory card concept in 1974. In 1977, Michel Ugon from Honeywell Bull invented the first microprocessor smart card.

Theory

A smart card, chip card, or integrated circuit card (ICC) is any pocket-sized card with embedded integrated circuits which can process data. This implies that it can receive input which is processed – by way of the ICC applications – and delivered as an output. There are two broad categories of ICCs. Memory cards contain only nonvolatile memory storage components, and perhaps some specific security logic. Microprocessor cards contain volatile memory and

microprocessor components. The card is made of plastic, generally PVC, but sometimes ABS. The card may embed a hologram to avoid counterfeiting. During the last decade, smart cards became available with new features such as a display or a keypad.

Two main categories of smart cards exist: contact cards, whereby the card and the card reader (which may be integrated in for instance a PC) are in contact during the operation/transaction, and ►contactless cards such as proximity and vicinity cards, where the card and the card reader communicate with each other over a short distance.

Smart cards and the related communication technologies used are standardized under ISO/IEC.

►EMV is the standard adopted by all major issuers of payment cards based on chip technology while the ►SIM/UICC is the smart card used for mobile communications.

Applications

Smart cards are an important enabler of e- and m-business applications, particularly because they can be used to hold authentication information such as a user's private key in a PKI infrastructure scheme or a user's biometric template. The card or card application may be activated/approved by the use of a cardholder verification method (CVM) such as a PIN or biometric sample, thus avoiding security issues associated with transmitting authentication credentials over computer networks. Smart cards may also be used in a wide variety of applications such as financial services, mobile services, transport, loyalty, authentication (e.g., for access control), and eID.

Recommended Reading

1. Guillou LC, Ugon M, Quisquater JJ (1992) The smart card, a standardised security device dedicated to public cryptography. In: Simmons G (ed) Contemporary cryptology – the science of information integrity. IEEE Press, New York, pp 561–613
2. Henry M (2007) Multi-application smart cards. Cambridge University Press, Cambridge
3. http://www.eurosmart.com/
4. http://globalplatform.org
5. http://www.smartcardalliance.org/
6. http://www.iso.org

Smart/Algorithmic Denial of Service

►Application-Level Denial of Service

Smartcard Tamper Resistance

Markus Kuhn
Computer Laboratory, University of Cambridge, Cambridge, UK

Related Concepts

►Compromising Emanations; ►Differential Power Analysis; ►Fault Attack; ►Side-Channel Analysis

Definition

►Tamper-resistant cryptographic modules are devices intended for applications that need to protect stored cryptographic keys and intermediate results of algorithms against unauthorized access.

Theory and Applications

The most popular portable form is the smartcard, which has the form of a banking plastic card with embedded microcontroller. The typical interfaces are either five visible electrical contacts (for ground, power supply, reset, clock, and a bidirectional serial port) or an induction loop. Typical smartcard processors are 8-bit microcontrollers with a few hundred bytes of RAM and 4–64 kilobytes of ROM or nonvolatile writable memory (NVRAM). Battery-like small steel cans ("crypto buttons"), CardBus/PCMCIA modules, and various PCI plug-in cards for non-portable applications are other popular form factors for tamper-resistant modules.

Smartcards are used in applications with both tamper-resistance and tamper-evidence requirements. Tamper resistance means that stored information must remain protected, even when the attacker can work on several samples of the module undisturbed for weeks in a well-equipped laboratory. Tamper evidence is a weaker requirement in which the regular holder of the module must merely be protected against unnoticed access to information stored in the module.

One common application for tamper-resistant smartcards are pay-TV conditional-access systems, where operators hand out millions of cards to customers, each of which contains the key necessary to descramble some subscription TV service. Pirates who manage to extract the key from one single issued card can use it to produce and sell illicit clone cards. Most proposed forms of digital rights management (DRM) mechanisms are based on some form of tamper-resistant element in the user system.

Examples for smartcard applications where operators can rely more on just a tamper-evidence requirement are ►digital signature identity cards, banking cards, and GSM

subscriber identity modules. Here, stored secrets are specific to a single card or cardholder and can be revoked, should the module get stolen.

There are four broad categories of attacks against tamper-resistant modules:

- *Software attacks* use the normal communication interface of the processor and exploit security vulnerabilities found in ►protocols, cryptographic algorithms, or the software implementation. Countermeasures involve very careful design and in-depth implementation reviews, possibly augmented by formal techniques.
- *Microprobing* techniques access the chip surface directly, such that the attacker is able to observe, manipulate, and interfere with the integrated circuit. This has been the dominant form of attack against pay-TV conditional-access cards since about 1993. Chemical depackaging (e.g., with fuming nitric acid) is used to dissolve conventional packaging materials without damaging the silicon chip. Microscopes with micromanipulators are then used to place fine tungsten hairs onto micrometer-wide on-chip bus lines, in order to establish an electrical contact between the chip circuits and recording equipment such as digital oscilloscopes. The glass passivation layer that covers the metal interconnects can be broken mechanically or removed with UV laser pulses. The content of the main memory can then be reconstructed from observed on-chip bus traffic, a process that can be simplified by damaging the instruction decoder to prevent the execution of jump commands. Attackers have also succeeded in accessing the memory with the help of circuitry placed on the chip by the manufacturer for postproduction testing. Modern chips with smaller feature sizes require the use of focused ion-beam workstations. With these, the surface of a depackaged chip can be modified inside a vacuum chamber. A beam of accelerated gallium ions and various added processing gases remove chip material or deposit either conducting and insulating substances with a resolution of tens of nanometers. This not only allows attackers to modify the metal connections between the transistors, effectively to edit the processor design, but also helps in establishing larger probing pads for the connection of recording equipment. Countermeasures involve more difficult-to-remove packaging materials (e.g., silicon, silicon carbide), obfuscated circuits, additional top-layer metal sensor meshes, the careful destruction of test circuitry before the chip is delivered to customers, and the design of instruction decoders that frustrate modifications aimed at simplifying access to all memory locations [1].
- *Fault generation* techniques or ►fault attacks use abnormal environmental conditions to generate malfunctions in the processor aimed at providing additional access. A simple example would be a deliberately caused and carefully timed glitch that disrupts the correct execution of a single security-critical machine instruction, such as the conditional branch at the end of a password comparison. Carefully placed, a single glitch can help to bypass many layers of cryptographic protection. Such glitches have been generated by increasing the provided clock frequency for a single cycle, by brief supply voltage fluctuations, by applying light flashes to the entire chip or single gates, and with the help of electromagnetic pulses. Another class of fault generation attacks attempts to reduce the entropy generated by hardware random-bit generators. For example, where multiple noisy oscillators are used to generate randomness, externally applied electromagnetic fields with carefully selected frequencies can result in a phase lock and more predictable output. Countermeasures against fault generation include adding filters into supply lines, regular statistical checks of random-bit generators, redundant consistency checks in the software, and new logic design techniques that lead to inherently glitch-resistant circuits.
- ►Eavesdropping or ►side-channel analysis techniques monitor with high time resolution the characteristics of all supply and interface connections and any other electromagnetic radiation produced by a processor. A simple example is the determination of the length of the correct prefix of an entered password from the runtime of the string-compare routine that rejects it. This can significantly reduce the average number of guesses needed to find the correct string. The nature of the executed instruction as well as parts of the processed data are evident in the power-supply current of a CPU. A conditional branch that takes effect can easily be distinguished from one that passes through by examining with an oscilloscope the voltage drop over a 10 Ω resistor inserted into the processor's ground connection line. The current consumed by the write operation into memory cells is often proportional to the number of bits that change their value. Even status register flags and Hamming weights of data processed in arithmetic units can show up in power consumption curves. The technique of ►differential power analysis determines secret-key bits by correlating measured current curves with externally simulated intermediate

results of a symmetric cipher. It has been demonstrated as a practical attack technique, even in situations where there has not been a microprobing attack first to disassemble the software in the targeted smartcard. Countermeasures include the addition of filters and shields against ►compromising emanations, circuitry, and routines for adding random noise and delays, new balanced or dual-rail logic design techniques that lead to inherently less information in the power signal, and algorithmic techniques for reducing the number of intermediate results useful for eavesdroppers.

Microprobing requires time and careful preparation in a laboratory environment and is therefore primarily a challenge of tamper resistance. The other three attack classes are noninvasive and can, with suitable preparation, be performed in just a few seconds with attack equipment that could be disguised as a regular smartcard reader. The holder of the card might not notice such an attack, and then even the tamper evidence would be lost.

Recommended Reading

1. Kömmerling O, Kuhn MG (1999) Design principles for tamper-resistant smartcard processors. In: Proceedings of the USENIX workshop on smartcard technology (Smartcard'99), 10–11 May 2009, Chicago, IL. USENIX Association, Berkeley, pp 9–20, ISBN 1-880446-34-0
2. Weingart SH (1965) Physical security devices for computer subsystems: a survey of attacks and defenses. Workshop on cryptographic hardware and embedded systems (CHES 2000). Lecture notes in computer science, vol 1965. Springer, Berlin, pp 302–317

Smoothness

Kim Nguyen
Bundesdruckerei GmbH, Berlin, Germany

Related Concepts

►Factorization; ►Index Calculus

Definition

A natural number n is called B-smooth if its factorization does not contain any prime factors larger than B, i.e.,

$$n = \prod_{p \le B} p^{n_p}.$$

Analogously, we can also consider elements of the polynomial ring $\mathbb{F}_p[x]$. For a polynomial f of degree $\deg(f)$ define the norm of f to be $p^{\deg(f)}$. An element of $\mathbb{F}_p[x]$ is called B-smooth if its factorization does not contain any irreducible polynomials of norm greater than B.

Theory

For $t, c \in \mathbb{R}$ such that $0 \le t \le 1$ the complexity–theoretic ►L-notation is defined by

$$L_x[t, \gamma] = e^{(\gamma + o(1))(\log x)^t (\log\log x)^{1-t}},$$

where $x \to \infty$. Note that for $t = 0$ this equals $(\log x)^\gamma$, while for $t = 1$ we obtain x^γ (neglecting the $o(1)$ term). Hence we see that for values of t between 0 and 1 the function L interpolates between ►polynomial time and ►exponential time behaviour. For these values we say that L is subexponential in x (►Subexponential time).

The main observation about the distribution of smooth numbers in an interval $[0, a]$ is that if the smoothness bound B is chosen subexponentially in x, then the probability that a random integer in this interval is B-smooth (or more precisely the inverse of that probability) is also subexponential.

More precisely, set $a = L_x[r, \alpha]$ and $B = L_x[s, \beta]$, where $r, s, \alpha, \beta \in \mathbb{R}_{>0}$ and $s < r \le 1$, then the probability that a random number in $[0, a]$ is B-smooth is given by

$$L_x[r - s, -\alpha(r - s)/\beta], \tag{1}$$

where $x \to \infty$.

A similar result holds for the polynomial case: Assume $r, s, \alpha, \beta \in \mathbb{R}_{>0}$ such that $r \le 1$ and essentially $s < r$. Then the probability that a random element of $\mathbb{F}_p[x]$ of norm bounded by $L_x[r, \alpha]$ is $L_x[s, \beta]$–smooth is given exactly by expression (1) (see [1] for details).

Smooth numbers or polynomials are used in the most effective methods to factor natural numbers (►Integer factoring) and compute discrete logarithms in finite fields (►Discrete logarithm problem). The overall subexponential complexity of these methods is a direct consequence of the fact that the number of smooth elements in a given interval grows subexponentially if the smoothness bound is chosen subexponential as well.

For further background, please see [2], [3] and [4].

Recommended Reading

1. Andrew M. Odlyzko Discrete logarithms in finite fields and their cryptographic significance. In: Beth T et al. (eds) pp 224–314
2. Canfield ER, Paul Erdős, Carl Pomerance (1983) On a problem of Oppenheim concerning "factorisatio numerorum". J Num Theory 17:1–28
3. de Bruijn NG (1951) On the number of positive integers ≤ x and free of prime factors > y. Indagationes Mathematicae 13:50–60
4. Ramaswami V (1949) The number of positive integers ≤ x and free of prime divisors > x_c, and a problem of S. S. Pillai. Duke Math J 16:99–109
5. Beth T, Cot N, Ingemarsson I (eds) (1985) Advances in cryptology: EUROCRYPT '84, vol 209. LNCS. Springer, Berlin

Social Perspectives on Information Privacy

Mary J. Culnan
Information and Process Management Department, Bentley University, Waltham, MA, USA

Related Concepts

►Macrodata Protection; ►Microdata Protection; ►Quasi-Identifier

Definition

Information privacy encompasses the rights and responsibilities of both individuals and organizations related to the collection, use, disclosure, and retention of personally identifiable information (PII).

Theory

Because privacy involves regulation of access to one's personal information by others, privacy is primarily a social rather than a technical issue. Sources of privacy include relief from social friction, the ability to shut out "the community," and the right not to participate in collective life. As a result, privacy is considered an essential element of democratic societies because it protects personal autonomy. However, privacy is not absolute; individual privacy interests need to be balanced with the legitimate information needs of society at large for accountability.

Privacy issues typically arise when organizations make new or unexpected uses of information they collected for one purpose for a different purpose, or from new technologies where norms lag deployment of the technology. As a result, we confront privacy issues on an ongoing basis in our roles as consumers, citizens, and employees.

As the definition above suggests, information privacy may be viewed from the dual perspectives of individuals and organizations. From the individual perspective, information privacy is a multidimensional concept that depends on context and also varies with a person's life experiences. Typically, it has been defined in terms of an individual's ability to control the disclosure and subsequent uses of their personally identifiable information (PII) [5]. PII is information that can be linked to an identifiable individual.

From the organizational perspective, information privacy involves assuring appropriate use of personal information through data protection. Data protection consists of rules that govern the collection, processing, and use of PII. Because privacy concerns typically result from organization's information processing activities, Daniel Solove [4] characterized privacy as a set of 12 issues arising from organizational information privacy practices. He developed a taxonomy of information processing and information dissemination activities that have the potential to result in a range of harms to individuals, thereby causing "privacy problems."

Most of these activities may be grouped into two broad categories: information reuse and unauthorized access to personal information. Consistent with Solove, security is defined as one aspect of privacy arising from unauthorized use of PII; however, privacy includes more than security. Security is about protecting PII, while privacy is broader and also encompasses permission and use of PII. Privacy is difficult to achieve without security. However, organizations can successfully secure the PII in their custody and still make bad decisions about how the PII they have collected is subsequently used, resulting in privacy problems. Both types of privacy problems, information reuse and unauthorized access, can potentially threaten individuals' ability to maintain a condition of limited access to their PII by others, harm individuals, and subsequently threaten an organization's legitimacy in its interactions with consumers, shareholders, and regulators [1].

Typically, information reuse involves organizations making legal decisions about new uses for the personal information they have collected. Examples of information reuse can include aggregation, data mining, or data sharing. These activities can result in organizations drawing incorrect inferences or making decisions based on errors, individuals being excluded from certain opportunities, or unwanted intrusions when PII is used for marketing without the individual's knowledge or permission.

Unauthorized access, the second category of privacy problem, includes two types of activities: browsing and data breaches. In the case of browsing, employees view PII they are not authorized to view such as the case of people who browse a celebrity's records. Breaches involve unauthorized access to personal information resulting from a variety of security incidents including hackers breaking into systems or networks, third parties accessing PII on lost laptops or other mobile devices, or organizations failing to dispose of PII securely. Privacy harms resulting from unauthorized access can include breach of confidentiality and trust, or financial harm to individuals resulting from identity theft or identity fraud.

Fair information practices (FIPs) provide a basis for organizations to create policies and processes that can avoid privacy problems. FIPs originated with the Organization for Economic Cooperation and Development's 1980 *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data* [3]. FIPs address privacy harms by

defining guidelines for responsible information use. While FIPs serve as the basis for privacy laws and industry self-regulatory programs around the world, their actual coverage and implementation varies, reflecting cultural and legal differences.

FIPs consist of three types of principles that define individual and organizational rights and responsibilities: knowledge, control, and stewardship. Knowledge includes collection limitation, notice, and openness. PII should be collected openly, fairly, and with the knowledge of the individual. Further, the purpose for collecting the individual's PII should be specified at the time of collection. Control includes a limitation on use and user participation. PII collected for one purpose should not be used for other purposes without the individual's consent, unless the use is required by law. People should have the right to know if their PII has been collected, to see their PII, and to correct errors. Stewardship requires organizations to be accountable for complying with the FIPs, to protect PII from unauthorized access, and to maintain data quality by ensuring that PII are relevant, accurate, and current.

FIPs may be implemented through a mix of laws, self-regulation, and technology. Throughout most of the developed world, countries have adopted omnibus national privacy laws, which apply both to the public and the private sectors. A national data protection or information privacy official is typically responsible for enforcing their country's privacy laws. One notable exception to this trend is the USA, which has adopted a sectoral approach to regulating privacy. Here, privacy is regulated through a mix of laws targeting specific problems and industry self-regulation resulting in gaps in coverage. Further, there is no national official in the USA with overall responsibility for privacy.

Open Problems and Future Directions

While the principles underlying FIPs are straightforward, their implementation poses challenges. For the most part, the key concepts lack definitions. For example, what constitutes effective notice? Notice is at the heart of individual control, but it is very difficult to write a privacy notice that is complete, readable, and understandable. How should notice be delivered to users in new or nontraditional information environments such as mobile computing? What new or additional uses require consent and how should consent be obtained? Should access be provided only for information the individual provided, or for information the organization acquired from third parties or derived? What constitutes effective security? The emerging legal regime requires "reasonable security," which varies according to the size of the organization and the sensitivity of PII in their custody. How do organizations ensure they comply with FIPs? Further, information practices vary across different business models and with time, and individuals vary in terms of which information practices they find valuable or objectionable. New technologies raise privacy issues that challenge existing privacy norms. Further, the laws governing the implementation of FIPs may impose requirements that are unclear or may vary across jurisdictions or types of data, or be in conflict with one another, thereby posing difficult compliance challenges, particularly for organizations operating globally.

As a result, privacy is often difficult for organizations and individuals and as a result, society [2]. This situation is likely to continue for the foreseeable future as organizations will continue to find new uses for PII and to adopt technologies which raise unanticipated privacy issues. Privacy, therefore, is likely to remain a moving target for individuals and organizations alike.

Recommended Reading

1. Culnan MJ, Williams CC (2009) How ethics can enhance organizational privacy: lessons from the ChoicePoint and TJX data breaches. MIS Quarterly 33(4):673–687
2. Marguiles ST (ed) (2003) Contemporary perspectives on privacy: social, psychological, political. J Soc Issues 59(2)
3. Organization for Economic Cooperation and Development (1980) Guidelines on the protection of privacy and transborder flows of personal data. Available at: http:www.oecd.org.
4. Solove D (2008) Understanding privacy. Harvard University Press, Cambridge
5. Westin AF (1967) Privacy and freedom. Atheneum, New York

Software-Based Attestation

►Sensor Code Attestation

Software-Optimized Encryption Algorithm

►SEAL

Solitaire

Bruce Schneier
BT, London, UK

Related Concepts

►Stream Cipher

Definition

Solitaire is a *stream cipher* designed to be implemented using a deck of cards.

Background

Solitaire was invented by Bruce Schneier for use in the novel *Cryptonomicon*, by Neal Stephenson [1], where it was called Pontifex. Solitaire gets its security from the inherent randomness in a shuffled deck of cards. By manipulating this deck, a communicant can create a string of "random" letters which he then combines with his message. Solitaire can be simulated on a computer, but it is designed to be used by hand.

Manual ciphers are intended to be used by spies in the field who do not want to be caught carrying evidence that they send and receive encrypted messages. In David Kahn's book *Kahn on Codes* [2], he describes a real pencil-and-paper cipher used by a Soviet spy. Both the Soviet algorithm and Solitaire take about the same amount of time to encrypt a message: most of an evening.

Solitaire, as described in the appendix to *Cryptonomicon*, has a cryptographic weakness. While this weakness does not affect the security of short messages, Solitaire is not recommended for actual use.

Recommended Reading

1. Stephenson N (2002) Cryptonomicon. Avon Eos Books, Avon
2. Kahn D (1984) Kahn on codes: secrets of the new cryptology. Macmillan, London

Source Location Privacy

Wensheng Zhang
Department of Computer Science, College of Liberal Arts and Sciences, Iowa State University, Ames, IA, USA

Related Concepts

►Anonymity; ►Source Location

Definition

Source location privacy in sensor networks means the status that the information about the locations of events detected by sensor nodes is properly protected such that only authorized entities, for example, the sink of the network, can obtain the information.

Background

Monitoring and detecting events is a typical application of sensor networks. When a sensor node detects an event, if it sends a report including the location of the event to the sink, the location of event source could be exposed to the adversary no matter how strong the data encryption key is, because the adversary may be passively monitoring the network. Preserving source location in sensor networks is challenging mainly due to the following reasons: the limited resources available in sensor networks require highly efficient privacy preservation mechanisms; the open nature of the underlying wireless communication makes each of the adversary to monitor or eavesdrop communications between sensor nodes [10].

Theory and Applications

Most of existing techniques for source location privacy preservation mainly adopt the random walk mechanisms, the dummy data mechanisms, and/or the fake data sources mechanisms [1, 2, 4, 6].

Ozturk et al. [3, 7] proposes phantom routing, an instance of the random walk mechanism. The idea is that, data sent by the event source first takes a few stops of random walk, and then is transmitted toward the sink via probabilistic flooding. This scheme is designed for the problem that only a single source is under a single attacker's consideration; the attacker has limited communication range and tries to trace back to the source in a hop-by-hop manner. As a pure random walk approach tends to stay around the real source, and hence is not statistically secure [9], Xi et al. [9] proposed the greedy random walk (GROW) scheme. GROW is a two-way random walk. Specifically, the sink first initiates a random walk, and the nodes on this random walk path are called receptors. Then, data from the event source also takes a random walk, and the walk ends until it hits a receptor. At that point, the data is forwarded to the sink along the path of random walk initiated by the sink. Osturk et al. [3, 7] also proposes a short-lived fake source strategy and a persistent fake source strategy, both of which inject dummy data into the network to increase the difficulty of identifying event source. With the short-lived fake source strategy, after receiving a real packet, each sensor sends out a fake packet with a predefined probability, such that the fake source changes from one fake message to another. With the persistent fake source strategy, each node has a predetermined probability to act as a fake source permanently.

To defend against stronger, global adversaries which can monitor the transmission rate of each sensor node, Shao et al. [8] proposed that every node in the network sends out dummy messages with intervals following a certain kind of distribution, and a node detecting a real event should transmit the real event message with intervals following the same distribution. This scheme introduces a

trade-off between extra communication overhead caused by dummy messages and the delay in transmitting real event messages: the lower the overhead the higher the delay. To address this issue, dummy packets are transmitted with the intervals following the exponential distribution, and when a real message needs to be sent, the shortest interval that can still keep the intervals follow the same distribution (i.e., make the intervals to pass the Anderson-Darling Test successfully) is found to send out the real message so as to make transmission delay as low as possible. They also proposed two other schemes, the proxy-based filtering scheme (PFS) and the tree-based filtering scheme (TFS) [10] to filter dummy traffic without scarifying privacy preservation, and hence to minimize extra communication overhead. Specifically, the PFS selects some sensors as proxies to collect all messages and filter dummy ones, and a heuristic algorithm was developed to place the proxies to minimize traffic overhead, while the TFS organizes proxies into a tree hierarchy.

Mehta et al. [5] proposed to create multiple candidate traces in the network to hide the traffic generated by real objects. In this approach, a set of virtual objects are simulated in the field, where each of them generates a traffic pattern similar to that of a real object. Such approach provides trade-offs between privacy, communication cost, and delay.

Open Problems

Most of existing source privacy preservation schemes mainly consider outsider attacks. As sensor nodes are easily compromised by strong adversaries, it remains a major challenge how to protect source privacy against both outsider and insider attacks.

Recommended Reading

1. Deng J, Han R, Mishra S (2004) Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. Proceedings of the international conference on dependable systems and networks (DSN), pp 637–646
2. Hoh B, Gruteser M (2005) Protecting location privacy through path confusion. Proceedings of the international conference on security and privacy for emerging areas in communications networks (SECURECOMM), pp 194–205
3. Kamat P, Zhang Y, Trappe W, Ozturk C (2005) Enhancing source-location privacy in sensor network routing. Proceedings of the IEEE international conference on distributed computing systems (ICDCS), pp 599–608
4. Li N, Zhang N, Das S, Thuraisingham B (2009) Privacy preservation in wireless sensor networks: a state-of-the art survey. Elsevier Ad Hoc Networks 7:1501–1514
5. Mehta K, Liu D, Wright M (2007) Location privacy in sensor networks against a global eavesdropper. Proceedings of the IEEE international conference on network protocols (ICNP), pp 314–323
6. Ouyang Y, Le Z, Chen G, Ford J, Makedon F (2006) Entrapping adversaries for source protection in sensor networks. Proceedings of the international symposium on world of wireless, mobile and multimedia networks (WoWMoM), pp 23–34
7. Ozturk C, Zhang Y, Trappe W (2004) Source-location privacy in energy-constrained sensor network routing. Proceedings of the ACM workshop on security of ad hoc and sensor networks (SASN), pp 88–93
8. Shao M, Yang Y, Zhu S, Cao G (2007) Towards statistically strong source anonymity for sensor networks. Proceedings of the IEEE international conference on computer communications (INFOCOM), pp 466–474
9. Xi Y, Schwiebert L, Shi W (2006) Preserving source location privacy in monitoring-based wireless sensor networks. Proceedings of the international parallel and distributed processing symposium (IPDPS)
10. Yang Y, Shao M, Zhu S, Urgaonkar B, Cao G (2008) Towards event source unobserability with minimum network traffic in sensor networks. Proceedings of the ACM conference on wireless network security (WiSec), pp 77–88

Space-Time Trade-Off

►Time-Memory Trade-offs

Spam Detection Using Network-Level Characteristics

Brent Byung Hoon Kang[1], Gautam Singaraju[2]
[1]Department of Applied Information Technology and Centre for Secure Information Systems, The Volgenau School of Engineering, George Mason University, Fairfax, VA, USA
[2]The Volgenau School of Engineering and IT, George Mason University Ask.com

Synonyms

Antispam based on sender reputation

Definition

Spam Detection Using Network-Level Characteristics is a type of method that is used to determine whether a given email is unsolicited based on the network characteristics of the sender, for example, the IP address, pattern of SMTP transactions, and the associated Autonomous System Number (ASN).

Background

Unsolicited email, also referred to as Spam, is one of the pernicious problems on Internet today. There have been numerous efforts to design an automated mechanism to

determine whether an email is unsolicited or not. These methods can have two flavors: One that is based on the email's content, and the other based on the network-level characteristics of the sender.

Theory and Applications

Senders could be identified by sender's email ID, IP address, or domain. For instance, PGP [4, 9] is an email ID-based authentication technique where a third-party server maintains individual users' public keys. A sender's signed emails can be verified by retrieving the sender's public key.

Blacklist IP and Real-time Blackhole List (RBL) [6] employ IP addresses to identity spammers that keep a list of IP addresses that propagate spam. Though several RBLs are available, a recent study has shown that only 50% of spam is correctly identified by combined use of two or more lists [5].

To identify a valid sender, sender identification techniques such as SenderID [2], Sender Policy Framework (SPF) [7], Domain Key Identified Mail (DKIM) [1, 3] and DomainKeys [8] have been developed. SPF verifies if the sender's email server is authorized to send email for the sender's domain. SenderID differs from SPF in its ability to authenticate either the "envelope from" header or the "purported responsible address." Using DKIM and DomainKeys, senders publish a set of public keys as a part of DNS; emails are then signed by their mail server. Receivers verify the signature, and hence, the sender.

A sender identity, either IP addresses or SPF/DKIM, enables a better spam filtering system [5]. It has also been noted that any spam detection system based on the sender identity larger than that of a single IP address can expose bad behavior more accurately than those based on observations of a single IP address. Network-level properties of spam senders can be embedded into spam filters and may be quite effective.

Recommended Reading

1. Allman E (2005) Domain Keys Identified Mail (DKIM): introduction and overview
2. Microsoft Corporation (2004) Sender id framework – executive overview
3. Peterson P (2006) SIDF and DKIM overview scorecard, authentication summit II
4. Price W (2003) Inside PGP key reconstruction. A PGP corporation white paper
5. Ramachandran A, Feamster N (2006) Understanding the network-level behavior of spammers. In: Proceedings of ACM SIGCOMM'06, Pisa, 11–16 Sept 2006
6. Realtime Blackhole List (2002) Mail Abuse Prevention System LLC, California. http://www.mail-abuse.org/rbl/
7. Wong MW (2004) Sender authentication: what to do, Technical document
8. Yahoo Inc, Domain Keys: proving and protecting email sender identity
9. Zimmermann P (1995) The official PGP user's guide. MIT Press, Cambridge

Speaker Biometrics

►Speaker Recognition

Speaker Identification and Verification (SIV)

►Speaker Recognition

Speaker Recognition

Homayoon Beigi
Research, Recognition Technologies, Inc., Yorktown Heights, NY, USA

Synonyms

Speaker biometrics; Speaker identification and verification (SIV); Talker recognition; Voice biometrics; Voice recognition; Voiceprint recognition

Definition

Speaker recognition is a multidisciplinary technology that uses the vocal characteristics of speakers to deduce information about their identities. It is a branch of biometrics that may be used for identification, verification, and classification of individual speakers, with the capability of tracking, detection, and segmentation by extension.

Background

In addressing the act of *speaker recognition* many different terms have been coined, some of which have caused great confusion. *Speech recognition* research has been around for a long time and, naturally, there is some confusion in the public between *speech* and *speaker* recognition. One term that has added to this confusion is *voice recognition*.

The term *voice recognition* has been used in some circles to double for *speaker recognition*. Although it is conceptually a correct name for the subject, it is recommended that the use of this term is avoided. *Voice recognition*, in the past, has been mistakenly applied to *speech recognition* and

these terms have become synonymous for a long time. In a speech recognition application, it is not the voice of the individual which is being recognized, but the contents of his/her speech. Alas, the term has been around and has had the wrong association for too long.

Other than the aforementioned, there have been a myriad of different terminology used to refer to this subject. These include *voice biometrics*, *speech biometrics*, *biometric speaker identification*, *talker identification*, *talker clustering*, *voice identification*, *voiceprint identification*, and so on. With the exception of the term *speech biometrics*, which also introduces the addition of a speech knowledge-base to speaker recognition, the rest do not present any additional information.

A human child develops an inherent ability to identify the voice of his/her parents before even learning to understand the content of their speech. In humans, speaker recognition is performed in the right (less dominant) hemisphere of the brain in conjunction with the functions for processing pitch, tempo, and other musical discourse. This is in contrast with most of the language functions (production and perception) in the brain, which are processed by the *Broca* and *Wernicke* areas in the left (dominant) hemisphere of the *cerebral cortex* [4].

A speaker recognition system first tries to model the vocal tract characteristics of a person. This may be a mathematical model of the physiological system producing the human speech or simply a statistical model with similar output characteristics as the human vocal tract. Once a model is established and has been associated with an individual, new instances of speech may be assessed to determine the likelihood of them having been generated by the model of interest in contrast with other observed models. This is the underlying methodology for all speaker recognition applications. The earliest known papers on speaker recognition were published in the 1950s [25, 31]. Initial speaker recognition techniques relied on a human expert examining representations of the speech of an individual and making a decision on the person's identity by comparing the characteristics in this representation with others. The most popular representation was the *formant* representation. In the recent decades, fully automated speaker recognition systems have been developed and are in use [4].

As for the importance of speaker recognition, it is noteworthy that *speaker identity* is the only biometric that may be easily tested (identified or verified) remotely through the existing infrastructure, namely the telephone network. This makes speaker recognition quite valuable and unrivaled in many real-world applications. It need not be mentioned that with the growing number of cellular (mobile) telephones and their ever-growing complexity, speaker recognition will become more popular in the future.

Speaker Enrollment

The first step required in most manifestations of speaker recognition is to enroll the users of interest. This is usually done by building a mathematical model of a sample speech from the user and storing it in association with an identifier. This model is usually designed to capture statistical information about the nature of the audio sample and is mostly irreversible – namely, the enrollment sample may not be reconstructed from the model.

Speaker Verification (Authentication)

In a generic speaker verification application, the person being verified (known as the test speaker), identifies himself/herself, usually by nonspeech methods (e.g., a username or an identification number). The provided ID is used to retrieve the enrolled model for that person, which has been stored according to the enrollment process, described earlier, in a database. This enrolled model is called the *target speaker model* or the *reference model*. The speech signal of the test speaker is compared against the target speaker model to verify the test speaker.

Of course, comparison against the target speaker's model is not enough. There is always a need for contrast when making a comparison. Therefore, one or more competing models should also be evaluated to come to a verification decision. The competing model may be a so-called (universal) background model or one or more cohort models. The final decision is made by assessing whether the speech sample given at the time of verification is closer to the target model or to the competing model(s). If it is closer to the target model, then the user is verified and otherwise rejected.

The speaker verification problem is known as a one-to-one comparison since it does not necessarily need to match against every single person in the database. Therefore, the complexity of the matching does not increase as the number of enrolled subjects increases. Of course, in reality, there is more than one comparison for speaker verification, as stated – comparison against the target model and the competing model(s).

Speaker Identification

There are two different types of speaker identifications, *closed-set* and *open-set*. Closed-set identification is the simpler of the two problems. In close-set identification, the

audio of the test speaker is compared against all the available speaker models and the speaker ID of the model with the closest match is returned. In practice, usually, the top best matching candidates are returned in a ranked list, with corresponding confidence or likelihood scores. In closed-set identification, the ID of one of the speakers in the database will always be closest to the audio of the test speaker; there is no rejection scheme.

One may imagine a case where the test speaker is a 5-year-old child where all the speakers in the database are adult males. In closed-set identification, still, the child will match against one of the adult male speakers in the database. Therefore, closed-set identification is not very practical. Of course, like anything else, closed-set identification also has its own applications. An example would be a software program that would identify the audio of a speaker so that the interaction environment may be customized for that individual. In this case, there is no great loss by making a mistake. In fact, some match needs to be returned just to be able to pick a customization profile. If the speaker does not exist in the database, then there is generally no difference in what profile is used, unless profiles hold personal information, in which case rejection will become necessary.

Open-set identification may be seen as a combination of closed-set identification and speaker verification. For example, a closed-set identification may be conducted and the resulting ID may be used to run a speaker verification session. If the test speaker matches the target speaker based on the ID, returned from the closed-set identification, then the ID is accepted and passed back as the true ID of the test speaker. On the other hand, if the verification fails, the speaker may be rejected altogether with no valid identification result. An open-set identification problem is therefore at least as complex as a speaker verification task (the limiting case being when there is only one speaker in the database) and most of the time it is more complex. In fact, another way of looking at verification is as a special case of open-set identification in which there is only one speaker in the list. Also, the complexity generally increases linearly with the number of speakers enrolled in the database since, theoretically, the test speaker should be compared against all speaker models in the database – in practice this may be avoided by tolerating some accuracy degradation [5].

Speaker and Event Classification

The goal of classification is a bit more vague. It is the general label for any technique that pools similar audio signals into individual bins. Some examples of the many classification scenarios are gender, age, and event classifications. Gender classification, as is apparent from its name, tries to separate male and female speakers. More advanced versions also distinguish children and place them into a separate bin; classifying male and female is not so simple in children since their vocal characteristics are quite similar before the onset of puberty. Classification may use slightly different sets of features from those used in verification and identification, depending on the problem at hand. Also, either there may be no enrollment or enrollment may be done differently [4].

Although these methods are called speaker classification, sometimes, the technique are used for event classification, such as classifying speech, music, blasts, gun shots, screams, whistles, horns, etc. The feature selection and processing methods for classification are mostly dependent on the scope and could be different from mainstream speaker recognition.

Speaker Segmentation, Diarization, Detection, and Tracking

Automatic segmentation of an audio stream into parts containing the speech of distinct speakers, music, noise, and different background conditions has many applications. This type of segmentation is elementary to the practical considerations of speaker recognition as well as speech and other audio-related recognition systems. Different specialized recognizers may be used for the recognition of distinct categories of audio in a stream.

An example is the ever-growing teleconferencing application. In a teleconference, usually, a host makes an appointment for a conference call and notifies attendees to call a telephone number and to join the conference using a special access code. There is an increasing interest from the involved parties to obtain transcripts (minutes) of these conversations. In order to fully transcribe the conversations, it is necessary to know the speaker of each statement. If an enrolled model exists for each speaker, then prior to identifying the active speaker (*speaker detection*), the audio of that speaker should be segmented and separated from adjoining speakers. When speaker segmentation is combined with speaker identification and the resulting index information is extracted, the process is called *speaker diarization*. In case one is only interested in a specific speaker and where that speaker has spoken within the conversation (the time stamps), the process is called *speaker tracking*.

Speaker Verification Modalities

There are two major ways in which speaker verification may be conducted. These two are called the *modalities* of speaker verification and they are *text-dependent* and *text-independent*. There are also variations

of these two modalities such as *text-prompted*, *language-independent text-independent*, and *language-dependent text-independent*.

In a purely *text-dependent* modality, the speaker is required to utter a predetermined text at enrollment and the same text again at the time of verification. Text dependence does not really make sense in an identification scenario. It is only valid for verification. In practice, using such text-dependent modality will be open to *spoofing* attacks; namely, the audio may be intercepted and recorded to be used by an impostor at the time of the verification. Practical applications that use the text-dependent modality do so in the text-prompted flavor. This means that the enrollment may be done for several different textual contents and at the time of verification, one of those texts is requested to be uttered by the test speaker. The chosen text is the prompt and the modality is called *text-prompted*.

A more flexible modality is the *text-independent* modality, in which case the texts of the speech at the time of enrollment and verification are completely random. The difficulty with this method is that because the texts are presumably different, longer enrollment and test samples are needed. The long samples increase the probability of better coverage of the idiosyncrasies of the person's vocal characteristics.

The general tendency is to believe that in the text-dependent and text-prompted cases, since the enrollment and verification texts are identical, they can be designed to be much shorter. One must be careful, since the shorter segments will only examine part of the dynamics of the vocal tract. Therefore, the text for text-prompted and text-dependent engines must still be designed to cover enough variation to allow for a meaningful comparison.

The problem of spoofing is still present with text-independent speaker verification. In fact, any recording of the person's voice should now get an impostor through. For this reason, text-independent systems would generally be used with another source of information in a multifactor authentication scenario.

In most cases, *text-independent* speaker verification algorithms are also *language-independent* since they are concerned with the vocal tract characteristics of the individual, mostly governed by the shape of the speaker's vocal tract. However, because of the coverage issue discussed earlier, some researchers have developed text-independent systems that have some internal models associated with phonemes in the language of their scope. These techniques produce a text-independent, but somewhat language-dependent speaker verification system. The language limitations reduce the space and, hence, may reduce the error rates.

Knowledge-Based Speaker Recognition (Speech Biometrics)

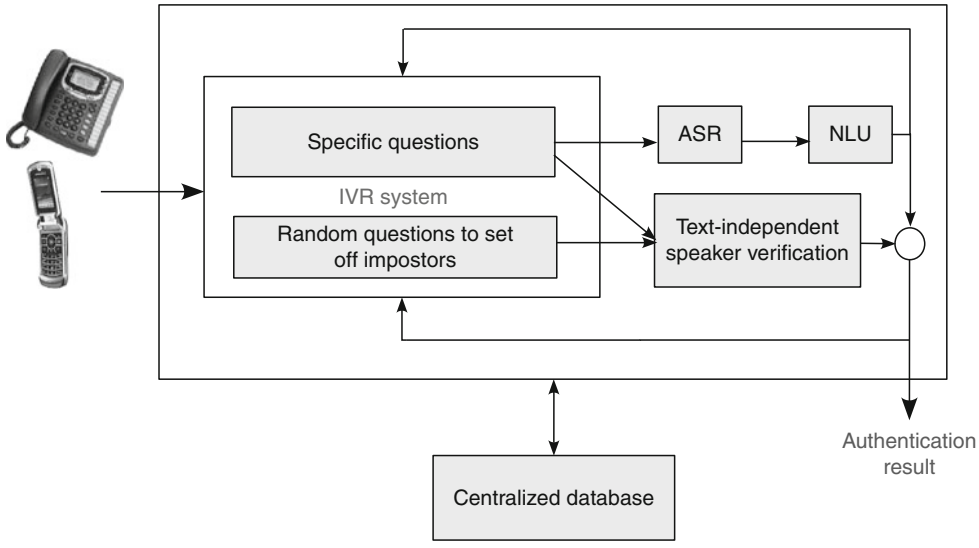
A knowledge-based speaker recognition system is usually a combination of a speaker recognition system and a speech recognizer and sometimes a natural language understanding engine or more. It is somewhat related to the *text-prompted* modality with the difference that there is another abstraction layer in the design. This layer uses knowledge from the speaker to test for liveness or act as an additional authentication factor. As an example, at the enrollment time, specific information such as a personal identification number (PIN) or other private data may be stored about the speakers. At the verification time, randomized questions may be used to capture the test speaker's audio and the content of interest. The content is parsed by doing a transcription of the audio and using a natural language understanding [20] system to parse for the information of interest. This will increase the factors in the authentication and is usually a good idea for reducing the chance of successful impostor attacks – see Fig. 1.

Theory

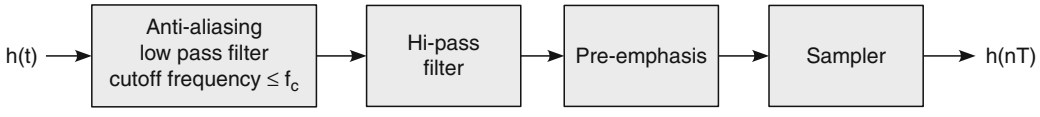
Speaker recognition is a multidisciplinary science. In its theory and implementation, it has a great deal in common with speech recognition [28]. It is impossible to cover the theory in this limited venue. The following disciplines are directly relevant: *signal processing*, *phonetics and phonology*, *information theory*, *Bayesian statistics and learning*, *optimization theory*, *parameter estimation*, *artificial intelligence*, and *applied mathematics*. Reference [4] provides a comprehensive coverage of the theory. An attempt is made here to list the different techniques that are used for speaker recognition.

As mentioned, the first step is to store the vocal characteristics of the speakers in the form of speaker models in a database for future reference. First, the features should be defined such that they would best represent the vocal characteristics of the speaker of interest. The most prevalent features used in the field happen to be identical to those used for speech recognition, namely, Mel Frequency Cepstral Coefficients (MFCCs).

Before extracting features, the audio signal should be sampled and made available with a fixed frequency that is determined based on the sampling theorem such that most of the information in the speech sample is preserved. There are many aspects to consider when the speech is sampled and stored in a standard format to be used by the speaker recognition engine. Figure 2 shows a typical sampling process that starts with an analog signal and produces a series of discrete samples at a fixed frequency, representing the speech signal. The discrete samples are usually



Speaker Recognition. Fig. 1 A practical speaker recognition system utilizing speech recognition and natural language understanding



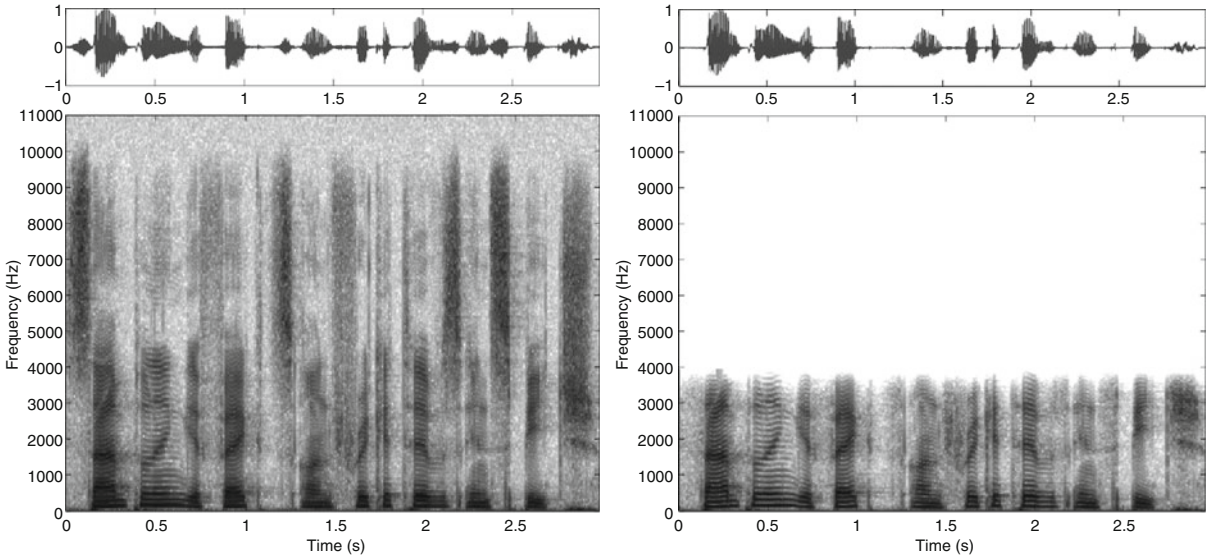
Speaker Recognition. Fig. 2 Block diagram of a typical sampling process

stored using a Codec (Coder/Decoder) format such as linear PCM, MU-Law, A-Law, etc. Standardization is quite important for interoperability of different engines [4]. The system of Fig. 2 should be designed so that it reduces *aliasing*, *truncation*, and *band-limitation* by choosing the right parameters such as the sampling rate and volume normalization. Figure 3 shows how most of the fricative information is lost going from a 22 kHz sampling rate to 8 kHz. Normal telephony sampling rates are at best 8 kHz. Mostly everyone is familiar with having to qualify fricatives on the telephone by using statements such as "S" as in "Sam" and "F" as in "Frank."

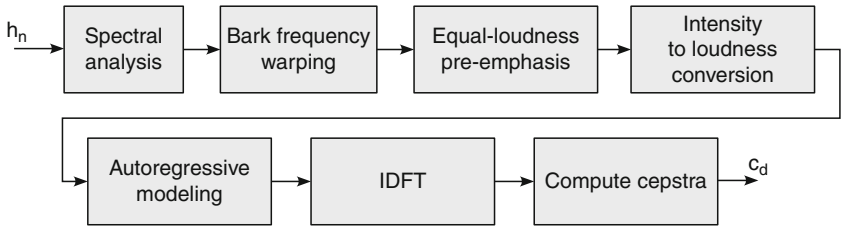
Cepstral coefficients have fallen out of studies in exploring the arrival of echos in nature [8]. They are related to the spectrum of the log of spectrum of a speech signal. The frequency domain of the signal in computing the MFCCs is warped to the Melody (Mel) scale. It is based on the premise that human perception of pitch is linear up to 1,000 Hz and then becomes nonlinear for higher frequencies (somewhat logarithmic). There are models of the human perception based on other warped scales such as the Bark scale. There are several ways of computing Cepstral coefficients. They may be computed using the direct method, also known as moving average (MA), which utilizes the fast Fourier transform (FFT) for the first pass and the discrete cosine transform (DCT) for the second pass to ensure real coefficients.

Some use the linear predictive, also known as autoregressive (AR) features by themselves: linear predictive coefficients (LPC), partial correlation (PARCOR) – also known as reflection coefficients, or log area ratios. However, mostly the LPCs are converted to cepstral coefficients using autocorrelation techniques. These are called linear predictive Cepstral coefficients (LPCCs). There are also the perceptual linear predictive (PLP) [18] features, shown in Fig. 4. PLP works by warping the frequency and spectral magnitudes of the speech signal based on auditory perception tests. The domain is changed from magnitudes and frequencies to loudness and pitch [4].

There have been an array of other features used such as *wavelet filterbanks* [9], for example, in the form of Mel-frequency discrete wavelet coefficients and wavelet octave coefficients of residues (WOCOR). There are also instantaneous amplitudes and frequencies that are in the form of amplitude modulation (AM) and frequency modulation (FM). These features come in different flavors such as



Speaker Recognition. Fig. 3 Utterance: "Sampling Effects on Fricatives in Speech," sampled at 22kHz (left) and 8kHz (right)



Speaker Recognition. Fig. 4 A typical perceptual linear predictive (PLP) system

empirical mode decomposition (EMD), FEPSTRUM, Mel Cepstrum modulation spectrum (MCMS) and so on [4].

It is important to note that most audio segments include a good deal of silence. Addition of features extracted from silent areas in the speech will increase the similarity of models, since silence does not carry any information about the speaker's vocal characteristics. Therefore, silence detection (SD) or voice activity detection (VAD) [4] is quite important for better results. Only segments with vocal signals should be considered for recognition. Other preprocessing such as audio volume estimation and normalization and echo cancellation may also be necessary for obtaining desirable result [4].

Once the features of interest are chosen, models are built based on these features to represent the speakers' vocal characteristics. At this point, depending on whether the system is text-dependent (including text-prompted) or text-independent, different methods may be used.

The models are tied to the type of learning that is done. A popular technique is the use of a Gaussian mixture model (GMM) [13] to represent the speaker. This is mostly relevant to the text-independent case which encompasses speaker identification and text-independent verification. Even text-dependent techniques can use GMMs, but, they usually use a GMM to initialize hidden Markov models (HMMs) [26] built to have an inherent model of the content of the speech as well. Many speaker diarization (segmentation and ID) systems use GMMs. To build a GMM of a speaker's speech, one should make a few assumptions and decisions. The first assumption is the number of Gaussians to use. This is dependent on the amount of data that is available and the dimensionality of the feature vectors. Standard clustering techniques are usually used for the initial determination of the Gaussians. Once the number of Gaussians is determined, some large pool of features is used to train these Gaussians (learn the parameters). This step is called training.

After the training is done, generally, the basis for a speaker-independent model is built. At this stage, depending on whether a universal background model (UBM) [30]

or cohort models are desired, different processing is done. For a UBM, a pool of speakers is used to optimize the parameters of the Gaussians as well as the mixture coefficients, using standard techniques such as maximum likelihood estimation (MLE), maximum a-posterioro (MAP) adaptation and maximum likelihood linear regression (MLLR). There may be one or more background models. For example, some create a single background model called the UBM, others may build one for each gender, by using separate male and female databases for the training. Cohort models are built in a similar fashion. A cohort is a set of speakers that have similar vocal characteristics to the target speaker.

At this point, the system is ready for performing the enrollment. The enrollment may be done by taking a sample audio of the target speaker and adapting it to be optimal for fitting this sample. This ensures that the likelihoods returned by matching the same sample with the modified model would be maximal.

At the identification and verification stage, a new sample is obtained for the test speaker. In the identification process, the sample is used to compute the likelihood of this sample being generated by the different models in the database. The identity of the model that returns the highest likelihood is returned as the identity of the test speaker. In identification, the results are usually ranked by likelihood. To ensure a good dynamic range and better discrimination capability, log of the likelihood is computed.

At the verification stage, the process becomes very similar to the identification process described earlier, with the exception that instead of computing the log likelihood for all the models in the database, the sample is only compared to the model of the target speaker and the background or cohort models. If the target speaker model provides a better log likelihood, the test speaker is verified and otherwise rejected. The comparison is done using the log likelihood ratio (LLR).

Of course, there are many other techniques used for the modeling of speakers. These include *neural networks*, kernel-based methods such as *support vector machines* [34], and other learning and classification theories. These techniques have also been combined with GMM models.

An extension of speaker recognition is diarization that includes segmentation followed by speaker identification and sometimes verification. The segmentation finds abrupt changes in the audio stream. Bayesian information criterion (BIC) [11] and generalized likelihood ratio (GLR) techniques and their combination [1] as well as other techniques [6] have been used for the initial segmentation of the audio. Once the initial segmentation is done, a limited speaker identification procedure allows for tagging the different parts with different labels. Figure 5 shows such a results for a two-speaker segmentation.

Speaker identification results are usually presented in terms of the error rate. They may also be presented as the error rate based on the result being present in the top N matches. This case is usually more prevalent in the cases where identification is used to prune a large set of speakers to only a handful of possible matches so that another expert system (human or machine) would finalize the decision process.

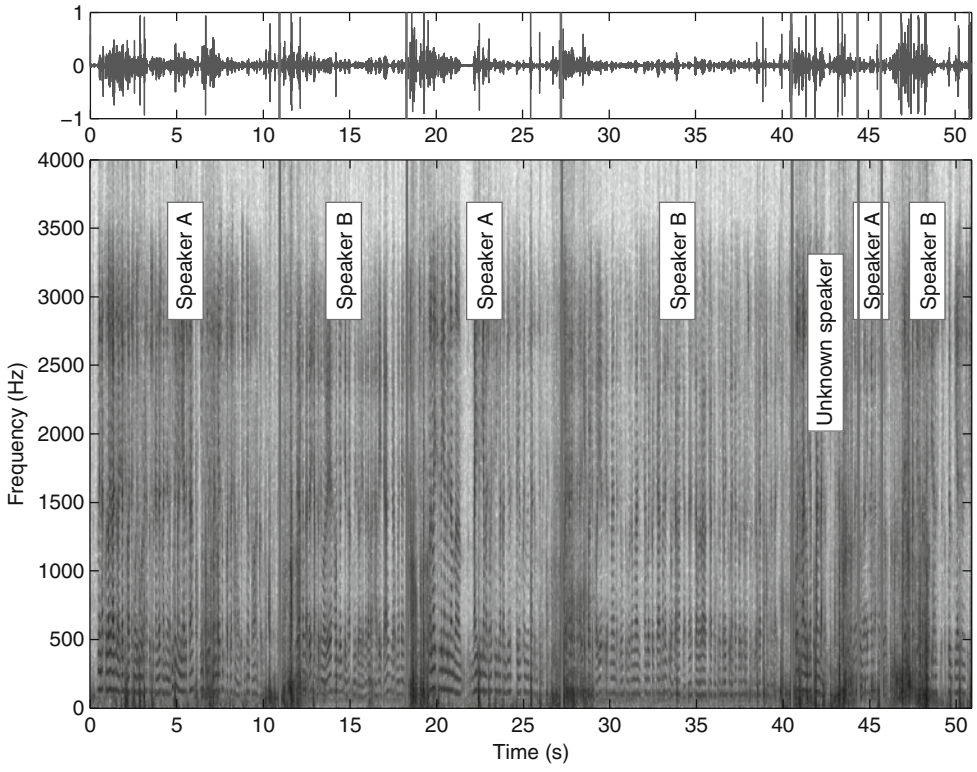
In the case of speaker verification, the method of presenting the results is somewhat more controversial. In the early days in the field, a *receiver operating characteristic (ROC)* curve was used [4]. For the past decade, the *detection error trade-off (DET)* curve [21, 22] has been more prevalent, with a measurement of the cost of producing the results, called the *detection cost function (DCF)* [22]. Figures 6 and 7 show sample DET curves for two sets of data underscoring the difference in performances. Recognition results are usually quite data-dependent. The next section will speak about some open problems which degrade results.

There is a controversial operating point on the DET curve that is usually marked as the point of comparison between different results. This point is called the *equal error rate (EER)* and signifies the operating point where the false rejection rate and the false acceptance rate are equal. This point does not carry any real preferential information about the "correct" or "desired" operating point. It is mostly a point of convenience that is easy to denote on the curve.

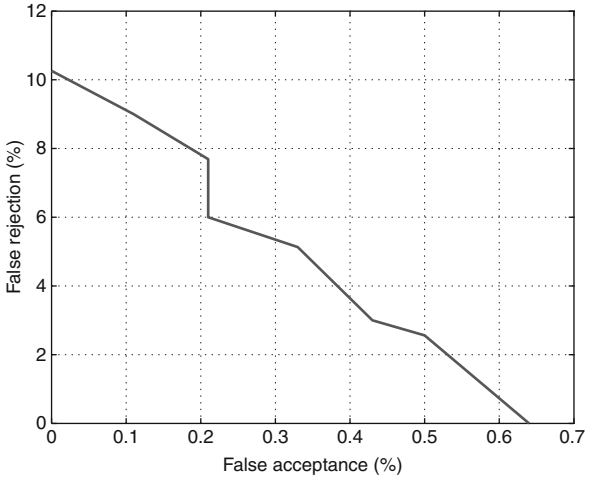
Applications

There are countless number of applications for the different branches of speaker recognition. These include, but are certainly not limited to, *financial*, *forensic, and legal* [23, 32] *access, control and security*; *audio/video indexing and diarization*; *surveillance*; *teleconferencing*; and *proctor-less distance learning*.

In designing a practical speaker recognition system, one should try to affect the interaction between the speaker and the engine to be able to capture as many vowels as possible. Vowels are periodic signals that carry much more information about the resonance subtleties of the vocal tract. In the text-dependent and text-prompted cases, this may be done by actively designing prompts that include more vowels. For text-independent cases, the simplest way is to require more audio in hopes that many vowels would be present. Also, when speech recognition and natural



Speaker Recognition. Fig. 5 Segmentation and labeling of two speakers in a conversation using turn detection followed by identification



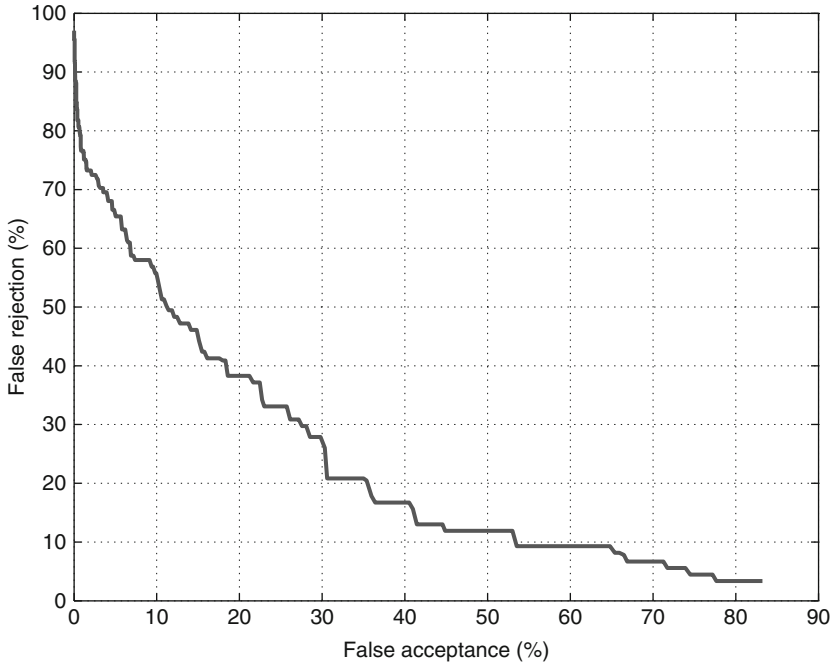
Speaker Recognition. Fig. 6 DET curve for quality data

language understanding modules are included (Fig. 1), the conversation may be designed to allow for higher vowel production by the speaker.

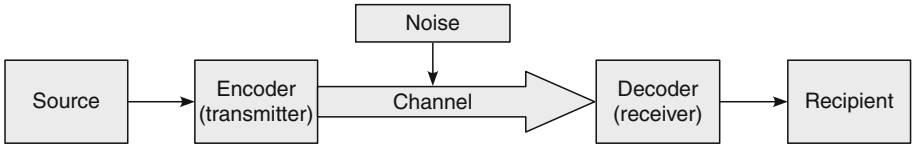
Open Problems

The greatest challenge in speaker recognition is the so-called channel-mismatch problem. Considering the general communication system given by Fig. 8, it is apparent that the channel and noise characteristics at the time of communication are modulated with the original signal. Removing these channel effects is the most important problem in information theory. This is of course a problem when the goal is to recognize the message being sent. It is, however, a much bigger problem when the quest is the estimation of the model that generated the message – as it is with the speaker recognition problem. In that case, the channel characteristics have mixed in with the model characteristics and their separation is nearly impossible. Once the same source is transmitted over an entirely different channel with its own noise characteristics, the problem of learning the source model becomes even harder.

Many techniques are used for alleviating this problem, but it is still the most important source of errors in speaker recognition. It is the reason why most systems that have been trained on a predetermined set of channels such



Speaker Recognition. Fig. 7 DET curve for highly mismatched and noisy data



Speaker Recognition. Fig. 8 One-way communication

as landline telephone could fail miserably when cellular (mobile) telephones are used. The techniques that are being used in the industry are listed here, but there are more techniques being introduced everyday:

- **Spectral filtering and Cepstral liftering**:
  - Cepstral mean subtraction (CMS) or Cepstral mean normalization (CMN) [7]
  - Cepstral mean and variance normalization (CMVN) [7]
  - Histogram equalization (HEQ) [12] and Cepstral histogram normalization (CHN) [7]
  - Autoregressive moving average (ARMA) [7]
  - Relative spectrAl (RASTA) filtering [17, 33]
  - J-RASTA [16]
  - Kalman filtering [19]
- **Other Techniques**
  - Vocal tract length normalization (VTLN) – first introduced for speech recognition: [10] and later for speaker recognition [15]
  - Feature warping [24]
  - Feature mapping [29]
  - Speaker model synthesis (SMS) [27]
  - Speaker model normalization [4]
  - H-norm (handset normalization) [14]
  - Z-norm and T-norm [2]

There are many challenges that have not been fully addressed in different branches of speaker recognition. For example, the large-scale speaker identification problem is one that is quite hard to handle. In most cases when researchers speak of large scale in the identification arena, they speak of a few thousand enrolled speakers. As the number of speakers increases to millions or even billions, the problem becomes quite challenging. As the number of speakers increases, doing an exhaustive match through the whole population becomes almost computationally implausible. Hierarchical techniques [5] would have to be utilized to handle such cases. In addition, the speaker space is really a continuum. This means that if one considers a

space where speakers who are closer in their vocal characteristics would be placed near each other in that space, then as the number of enrolled speakers increases, there will always be a new person that would fill in the space between any two neighboring speakers. Since there are intra-speaker variabilities (differences between different samples taken from the same speaker), the intra-speaker variability will be at some point more than inter-speaker variabilities, causing confusion and eventually identification errors. Since there are presently no large databases (in the order of millions and higher), there is no indication of the results, both in terms of the speed or processing and accuracy.

Another challenge is the fact that over time, the voice of speakers may change due to many different reasons such as illness, stress, aging, etc. One way to handle this problem is to have models which constantly adapt to changes [3].

Yet another problem plagues speaker verification. Neither background models nor cohort models are error-free. Background models generally smooth out many models and unless the speaker is considerably different from the norm, they may score better than the speaker's own model. This is especially true if one considers the fact that nature is usually Gaussian and that there is a high chance that the speaker's characteristics are close to the smooth background model. If one were to only test the target sample on the target model, this would not be a problem. But since a test sample that is different from the target sample (used for creating the model) is used, the intra-speaker variability might be larger than the inter-speaker variability between the test speech and the smooth background model.

There are, of course, many other open problems. Some of these problems have to do with acceptable noise levels until breakdown occurs. Using a cellular telephone with its inherently band-limited characteristics in a very noisy venue such as a subway (metro) station is one such challenging problem.

Given the number of different operating conditions in invoking speaker recognition, it is quite difficult for technology vendors to provide objective performance results. Results are usually quite data-dependent and different data sets may pronounce particular merits and downfalls of each provider's algorithms and implementation. A good speaker verification system may easily achieve an 0% EER for clean data with good inter-speaker variability in contrast with intra-speaker variability. It is quite normal for the same "good" system to show very high equal error rates under severe conditions such as high noise levels, bandwidth limitation, and small relative inter-speaker variability compared to intra-speaker variability. However, under most controlled conditions, equal error rates below 5% are readily achieved. Similar variability in performance exists in other branches of speaker recognition, such as identification, etc.

Recommended Reading

1. Ajmera J, McCowan I, Bourlard H (2004) Robust speaker change detection. IEEE Signal Process Letts 11(8):649–651
2. Auckenthaler R, Carey M, Lloyd-Thomas H (2000) Score normalization for text-independent speaker verification systems. Digital Signal Process 10(1–3):42–54
3. Beigi H (2009) Effects of time lapse on speaker recognition results. In 16th International Conference on Digital Signal Processing, pp 1–6, July 2009
4. Beigi H (2010) Fundamentals of speaker recognition. Springer, New York
5. Beigi HSM, Maes SH, Chaudhari UV, Sorensen JS (1999) A hierarchical approach to large-scale speaker recognition. In EuroSpeech 1999, Vol 5, pp 2203–2206, Sep 1999
6. Beigi HSM, Maes SH (1998) Speaker, channel and environment change detection. In: Proceedings of the World Congress on Automation (WAC 1998), May 1998
7. Benesty J, Sondhi MM, Huang Y (2008) Handbook of speech processing. Springer, New York
8. Bogert BP, Healy MJR, Tukey JW (1963) The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In: Rosenblatt M (ed) Time Series Analysis. Wiley, New York pp 209–243
9. Burrus CS, Gopinath RA, Guo H (1997) Introduction to wavelets and wavelet transforms: a primer. Prentice Hall, New York
10. Chau CK, Lai CS, Shi BE (2001) Feature vs. model based vocal tract length normalization for a speech recognition-based interactive toy. In Active Media Technology, Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp 134–143
11. Chen SS, Gopalakrishnan PS (1998) Speaker, environment and channel change detection and clustering via the Bayesian inromation criterion. In IBM Techical Report, T.J. Watson Research Center, New York
12. de la Torre A, Peinado AM, Segura JC, Perez-Cordoba JL, Benitez MC, Rubio AJ (2005) Histogram equalization of speech representation for robust speech recognition. IEEE Trans Speech Audio Process 13(3):355–366
13. Duda RO, Hart P (1973) Pattern classification and scene analysis. Wiley, New York
14. Dunn RB, Reynolds DA, Quatieri TF (2000) Approaches to speaker detection and tracking in conversational speech. Digital Signal Process 10:92–112
15. Grashey S, Geibler C (2006) Using a vocal tract length related parameter for speaker recognition. In the Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006. Pnerto Rico pp 1–5, Jun 2006
16. Hardt D, Fellbaum K (1997) Spectral subtraction and rasta-filtering in text-dependent hmm-based speaker verification. In IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol 2, pp 867–870
17. Hermansky H (1991) Compensation for the effect of the communication channel in the auditory-like analysis of speech (rasta-plp). In: Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH-91), pp 1367–1370

18. Hermansky H (1990) Perceptual linear predictive (plp) analysis of speech. J Acoust Soc Am 87(4):1738–1752
19. Kim NS (2002) Feature domain compensation of nonstationary noise for robust speech recognition. Speech Comm 37(3–4): 59–73
20. Manning CD (1999) Foundations of statistical natural language processing. The MIT Press, Boston
21. Martin A, Doddington G, Kamm T, Ordowski M, Przybocki M (1997) The det curve in assessment of detection task performance. In Eurospeech 1997. Rhodes, Greece pp 1–8
22. Martin A, Przybocki M (2000) The nist 1999 speaker recognition evaluation – an overview. Digital Signal Process 10:1–18
23. Nolan F (1983) The phonetic bases of speaker recognition. Cambridge University Press, New York
24. Pelecanos J, Sridharan S (2001) Feature warping for robust speaker verification. In A Speaker Odyssey - The Speaker Recognition Workshop, pp 213–218, Jun 2001
25. Pollack I, Pickett JM, Sumby WH (1954) On the identification of speakers by voice. J Acoust Soc Am 26:403–406
26. Poritz AB (1988) Hidden markov models: a guided tour. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP-1988), Vol 1, pp 7–13
27. Teunen R, Shahshahani B, Heck L (2000) A model-based transformational approach to robust speaker recognition. In International Conference on Spoken Language Processing, Vol 2, pp 495–498
28. Rabiner L, Juang B-H (1990) Fundamentals of speech recognition. Prentice Hall Signal Processing Series. PTR Prentice Hall, New Jersey
29. Reynolds DA (2003) Channel robust speaker verification via feature mapping. In the proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003 (ICASSP '03), Vol 2, pp II–53–6, Apr 2003
30. Reynolds DA, Quatieri TF, Dunn RB (2000) Speaker verification using adapted gaussian miscture models. Digital Signal Process 10:19–41
31. Shearme JN, Holmes JN (1959) An experiment concerning the recognition of voices. Lang Speech 2:123–131
32. Tosi OI (1979) Voice identification: theory and legal applications. University Park Press, Baltimore
33. van Vuuren S (1996) Comparison of text-independent speaker recognition methods on telephone speech with acoustic mismatch. In International Conference on Spoken Language Processing (ICSLP). Philadelphia, pp 784–787, Oct 1996
34. Vapnik VN (1998) Statistical learning theory. Wiley, New York

Special-Purpose Cryptanalytical Hardware

Christof Paar
Lehrstuhl Embedded Security, Gebaeude IC 4/132,
Ruhr-Universitaet Bochum, Bochum, Germany

Related Concepts

►FPGAs in Cryptography

Definition

Special-purpose cryptanalytical hardware refers to computing machines that were specifically designed to accelerate computations needed for cryptanalysis.

Background

Almost all symmetric and asymmetric cryptographic algorithms are merely computationally secure, that is, they can be broken if sufficient computing resources are available. In order to prevent successful attacks, the cryptographic parameters – which include, in particular, the key length – are chosen such that the required number of operations cannot be realized by an attacker. For instance, the key length of 128 bits for symmetric algorithms forces an attacker to perform 2^{128} encryptions for an exhaustive key search, which is entirely out of reach with current computing technology.

Theory

Cryptanalytical machines are special-purpose computers, which attempt to accelerate the computations involved in cryptographic attacks relatively to standard computers. One should bear in mind that most cryptanalytical machines do not change the mathematical complexity of an attack. Rather, the individual operations are accelerated. Cryptanalytical machines typically provide a speed-up factor in the range of 10–1,000 over clusters of conventional computers. Historically, cryptanalytical machines were mainly relevant in the context of code breaking by governments. A well-documented example is *Colossus*, an early programmable computer used by the British intelligence agencies during Word War II for breaking traffic encrypted with the Lorentz cipher machine. Also well known are the other World War II machines *Bomba*, built by the Polish intelligence agencies, and the *Bomb*, built by the British counterparts. Both greatly aided in the cryptanalysis of the Enigma ciphers.

With respect to modern ciphers, cryptanalytical machines were most often discussed in the context of DES. Already in 1977 it was claimed that a special-purpose machine, which would have cost tens of millions of US dollars, could perform an exhaustive-search attack against *Data Encryption Standard (DES)* [1]. A more detailed design, yet without any actual implementation, was presented in 1993 by Wiener [2]. In 1998 the Deep Crack computer, based on special-purpose integrated circuits, was able to perform an *exhaustive key search* of the DES as part of a code-breaking challenge. In 2006 the FPGA-based machine CO-PACOBANA was also able to brute-force attack DES, but at considerably lower costs [6]. It seems likely that government agencies had realized

DES-attack machines earlier. Other symmetric ciphers for which cryptanalytical machines were proposed include the GSM voice encryption cipher *A5/1*. Another application domain for such machines is password hashing in order to accelerate dictionary and brute-force attacks.

There have also been a few proposals for special-purpose machines for attacking asymmetric algorithms. The first attempt was the optoelectronic TWINKLE machine in 1999 that aids in factoring integers, the main operation when attacking RSA [3]. Subsequent proposals include *TWIRL*, another factorization computer based on standard CMOS technology [4]. There are no reports about actual realizations of attack machines against RSA. There have also been proposals for accelerating the Pollard Rho attack against elliptic curve cryptosystems [5, Chap. 6]. The recent availability of low-cost programmable platforms such as Graphic Processing Units (GPUs) and Cell Processors, which allow a very high number of arithmetic operations per time unit, are attractive for launching attacks against asymmetric ciphers. Hence, it can be speculated that custom-built hardware machines will become less attractive with respect to asymmetric cryptanalysis. However, special-purpose hardware can still have major advantages when attacking symmetric ciphers.

Applications

Since special-purpose hardware machines typically merely accelerate existing cryptanalytical algorithms by a constant factor, their main application domain is the attack of ciphers with a security of less than approximately 70 bits. Such algorithms are usually considered of not having an adequate security level. Symmetric algorithms with key lengths of more than 90 bits, such as *Rijndael/AES* and *triple DES*, asymmetric schemes based on *integer factoring* and the *discrete logarithm problem* with moduli of at least 2,048 bits, or *elliptic curve cryptography* with at least 192 bits, are not threatened by special-purpose hardware.

Recommended Reading

1. Diffie W, Hellman M (1977) Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84
2. Wiener MJ (1997) Efficient DES key search: an update. Cryptobytes 3(2):6–8
3. Shamir A (1999) Factoring large numbers with the TWINKLE device. In: CHES '99: Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems, vol 1717 of LNCS, pp 2–12, Springer, August 1999
4. Shamir A, Tromer E (2003) Factoring large numbers with the TWIRL device. In: Proceedings of Advances in Cryptology – Crypto 2003, vol 2729 of LNCS, pp 1–26. Springer, 2003
5. Güneysu T (2009) Cryptography and cryptanalysis on reconfigurable devices. PhD thesis, Department of Electrical Engineering and Computer Sciences, Ruhr-University Bochum, Germany, February 2009. http://www.crypto.ruhr-uni-bochum.de/en_theses.html
6. Güneysu T, Kasper T, Novotny M, Paar C, Rupp A (2008) Cryptanalysis with COPACOBANA. IEEE Trans Comput 57(11): 1498–1513

Specific Emitter Identification (SEI)

►Wireless Device Fingerprinting

Specific Emitter Verification (SEV)

►Wireless Device Fingerprinting

SPKI

Carl M. Ellison
NYC

SPKI (Simple ►Public Key Infrastructure) [1, 2] was developed starting in 1995 to remedy shortcomings [3] in the existing ID ►certificate definitions: ►X.509 and PGP (►Pretty Good Privacy). It provided the first authorization ►certificate definition [4, 5]. Originally, SPKI used no names for keyholders but, after the merger with SDSI (Simple Distributed Security Infrastructure), now includes both named keyholders and named groups or roles – specifying authorization grants to names and definitions of names (membership in named groups).

In public-key security ►protocols, the remote party (the *prover*) in a transaction is authenticated via ►public key cryptography. Upon completion of that ►authentication, the *verifier* has established that the prover has control over a particular private key – the key that corresponds to the public key the verifier used. This public key is itself a good identifier for the prover. It is a byte string that is globally unique. It also has the advantages of not requiring a central ID creator or distributor and of being directly usable for authentication. However, since anyone can create a key pair at any time, a raw public key has no security value. It is the purpose of a certificate to give value or meaning to this public key.

ID certificate systems bind names to public keys. This is an attempt to directly answer the question "who is that

other party?" The shortcomings of ID certificates that SPKI addresses are:

1. Because there is no single, global name source, names are not globally unique. Therefore, mapping from public key to name can introduce nonuniqueness. In SPKI, the real identifier is a public key or its cryptographic hash – each of which is globally unambiguous.
2. Names have no special value to a computer, but are strongly preferred by people over raw keys or hash values. However, people have a limited ability to distinguish from among large numbers of names, so the use of names can introduce scaling problems. The original SPKI did not use names, but SDSI names are defined by the *Relying Party (RP)* and presumably limited to the set that the RP can distinguish.
3. Name assignments are made by some ►Certificate Authority *(CA)* and the introduction of that additional component reduces overall system security. In SPKI/SDSI there is no CA, in the X.509 sense.
4. The real job is to make a security decision and a name by itself does not give enough information to make that decision. SPKI carries authorization information.

There are certain characteristics of SPKI/SDSI that set it apart from other certificate systems: SDSI names, authorization algebra, threshold subjects, canonical S-expressions, and ►certificate revocation (authorization architecture, authorization management, and authorization policy).

SDSI Names: Keys, and by implication their keyholders, need to be identified. SPKI uses the public key itself or its cryptographic hash as the ID of the key and the keyholder. This ID is globally unique and requires no issuer, therefore no expense or added insecurity of an ID issuer. For computers and the protocols between them, this ID is nearly perfect: globally unique and directly authenticable. The hash of the key has the added advantage of being fixed length.

For humans, such IDs fail miserably. They have no mnemonic value. SPKI uses SDSI names for human interfaces. Each human in a system using SPKI/SDSI maintains his or her own dictionary mapping between that human's preferred name for a keyholder and the public key or hash. The human operator can see friendly and meaningful names displayed via a UI, while the underlying system uses the key or its hash as an ID.

Source of Names

There is sometimes confusion among *X.509, PGP*, and SDSI – all of which build name certificates. The best way to distinguish them is via the source of the names used (see Table 1).

X.509 started out planning to use globally unique assigned names from the one global X.500 directory. That single directory has never been created and is unlikely ever to be. This leaves X.509 names to be chosen by the CA that issues a certificate. PGP leaves choice of name up to the person generating the key. SDSI gives choice of name to the person who will need to use that name.

Advantage of SDSI Names. When a name is used by a human, the correctness of that use depends on whether the human calls the correct person, thing, or group, to mind on seeing the name. When SDSI names are used, the one who chose that name is the same person who must correctly understand the linkage between the name and the person, thing, or group.

For example, the RP might choose the SDSI name "John Smith," if the RP knows only one John Smith – but a global naming authority would form "John Smith 3751" or ►jsmith39@localisp.net and require the RP to somehow deduce from that name which John Smith was intended. If the RP has an offline channel to John Smith and can ask him what his global ID is, then the RP can keep a local mapping from his preferred "John Smith" to the global name – but that is exactly what happens with SDSI (the global name being the hash of a key). If the RP does not have off-line contact with this John Smith, then the RP is forced to guess which John Smith is behind the name – and that guess is a source of security error [6].

Group Names

Both X.509 and PGP assume that the name is of an individual. SDSI names are of groups or roles. A named individual is a group of one.

Globally Unique SDSI Names

There are times when a SDSI name needs to be included in a certificate: when rights are assigned to the name or the name is added to some other named group. Since SDSI names are inherently local, a global form must be constructed. For example:

(name (hash sha 1 #14dc6cb49900bdd6d67f03f91741 cfefa2d26fa2#) Leanna)

SPKI. Table 1 Certificate name sources

Type	Source of names
X.509	Certificate Authority (CA)
PGP	End Entity (EE)
SDSI	Relying Party (RP)

stands for the name Leanna in the local dictionary of the keyholder of the key that hashes via SHA1 to 14dc6cb49900bdd6d67f03f91741cfefa2d26fa2.

This is an advantage of SPKI/SDSI over other ID certificate forms. SDSI knew that names were local and had to do something to make them globally unique while X.509 and PGP assumed names were global, even when they were not.

Authorization Algebra: SPKI carries authorization information in its ►certificates and (►Access Control List) ACL entries. This authorization is constrained to be in a language defined by SPKI so that the SPKI library can perform set intersections over authorizations. Each authorization is a set of specific permissions. That set is expressed as an enumeration (a literal set) or in a closed form (e.g., ranges of strings or numbers). The language is defined by intersection rules [2] that were designed to permit the intersection of two authorization sets to be expressed in the same closed form (authorization architecture, authorization management, and authorization policy).

By contrast, X.509v3 certificate extensions can be used to carry permission information, but because the extension is completely free-form, custom code must be written to process each different extension type.

Format: An SPKI certificate has five fields:

1. Issuer: the key of the certificate issuer.
2. Subject: a key, hash, SDSI name or threshold subject construct.
3. Delegation: a Boolean, indicating whether the Subject is allowed to delegate some or all of the rights granted here.
4. Tag: a canonical S-expression listing a set of rights granted by the issuer to the subject.
5. Validity: limits on validity: not-before or not-after dates, requirements to check online status or to get a revocation list, etc.

An SPKI ACL entry has fields 2.5 of the above since the authority (issuer) of an ACL entry is the machine that holds it.

A name membership certificate has four fields:

1. Issuer
2. Name being defined
3. Subject (key, hash or name)
4. Validity

There is one certificate for each member of a name.

A *threshold subject* is a list of N subjects (possibly including a subordinate threshold subject) and a parameter K. Only when K of the N subjects agree to delegate some rights or sign some document is that certificate or ACL entry considered valid. (The keys used by these subjects need not be in the same algorithm so, among other things, a threshold subject might tolerate the catastrophic break of one algorithm.)

Canonical S-Expressions (CSEXP)

SPKI/SDSI certificates are expressed and communicated as canonical S-expressions. An S-expression is of power equivalent to XML (Extensible Markup Language). Canonical S-expressions are binary forms with only one possible encoding. S-expressions in SPKI/SDSI are constrained to have each list start with an atom (the equivalent of an XML element name). Atoms are binary strings, with an explicit length stated, so CSEXP creation is trivial. CSEXP parsing requires less than 10 KB of code, in the open-source implementation. If element names are kept small, CSEXP binary forms are smaller than equivalent ASN.1 forms.

Certificate Revocation: At the time SPKI was designed, X.509 used *Certificate Revocation Lists (CRL)* that were optional and were not dated. A new CRL could be issued at any time and would override any prior CRL. In SPKI, revocation is deterministic. Each certificate that could be subject to revocation includes the revocation/validation agent's key and URL and all validity instruments (CRLs, etc.) have contiguous, nonoverlapping date ranges.

Implementations: SPKI certificates are used in HP's eSpeak and several prototype systems. It is available in open source code in two sourceforge.net projects: CDSA and JSDSI. SPKI's spiritual descendent XrML V.2 [5] is in use in Microsoft's Rights Management Services (RMS).

Recommended Reading

1. Ellison Carl SPKI/SDSI certificates. http://theworld.com/~cme/html/spki.html
2. Ellison Carl, Frantz Bill, Lampson Butler, Rivest Ronald, Thomas Brian, Ylönen Tatu (1999) SPKI certificate theory. IETF RFC2693, September 1999, ftp://ftp.isi.edu/in-notes/rfc2693.txt
3. Ellison Carl (2002) Improvements on conventional PKI wisdom. 1st Annual PKI Research Workshop, April 2002, http://www.cs.dartmouth.edu/~pki02/Ellison/
4. Blaze Matt KeyNote; http://www.crypto.com/trustmgt/kn.html
5. ISO/IEC JTC1/SC29/WG11/N5231XrML V.2 (MPEG-21 Rights Expression Language). http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm#_Toc23297977
6. Dohrmann Steve, Ellison Carl (2002) Public-key support for collaborative groups. In: 1st Annual PKI Research Workshop, April 2002, pp 139–148, http://www.cs.dartmouth.edu/~pki02/Dohrmann/

Spyware

Nary Subramanian
Department of Computer Science, The University of Texas at Tyler, TX, USA

Synonyms

Adware; Malware; Trojan

Related Concepts

►Keylogging; ►Permissions; ►Rootkits; ►Sandbox

Definition

Spyware is a software program running on a computing device that monitors the activities of the users of the device without the users' knowledge and reports that information to an external agency.

Background

Very frequently, in several computing devices including desktops, laptops, PDAs, and smartphones, software products have been detected running in the background, unknown to the users of these devices, that collect user data including their user names, passwords, credit card details, their contacts, other personal information, websites visited, personal habits such as frequently shopped for items, sports and movie preferences, and eating preferences. Usually such data are then sent, again unknown to the user, to an external agency which may then profit by selling this information to marketing agencies and others. Software products that perform this spying on behalf of external agencies are referred to as spyware. Studies have shown that almost 90% of consumer PCs have been infected with spyware and that on an average a PC has about 30 pieces of spyware [1]. Typically, home PCs have been infected more than industry PCs [2]. Several PC users have lost personal data through spyware and have faced subsequent financial losses as well when their bank accounts and passwords were fraudulently accessed by spyware customers [3]. While spyware has invaded PCs through free and purchased software, recently Internet has emerged as the leading source of spyware infections. In 2006, trojan horses infected 31% of PCs, system monitors infected 6% of PCs, and adware infected 59% of PCs. Cost to recover from spyware infection was US $100 per infection in 2007. Identity theft was reported by about 32% of spyware infected users and the total losses due to identity theft has been estimated by US Federal Trade Commission at $50 billion in 2006 [3]. Examples of spyware include Trojan-Downloader-Zlob which is a trojan horse downloader, Perfect Keylogger that records all internet sites visited besides all keystrokes and mousclicks, and FavoriteMan which is a browser helper object for Internet Explorer that connects to the internet, transmits your email address, and installs other programs. Most spyware seem to target devices operating on Microsoft's Windows operating systems since, as one theory goes [4], they are in the majority of PCs and laptops; however, spyware in the form of adware target users of all operating systems including Mac OS, Linux, and Windows.

Theory

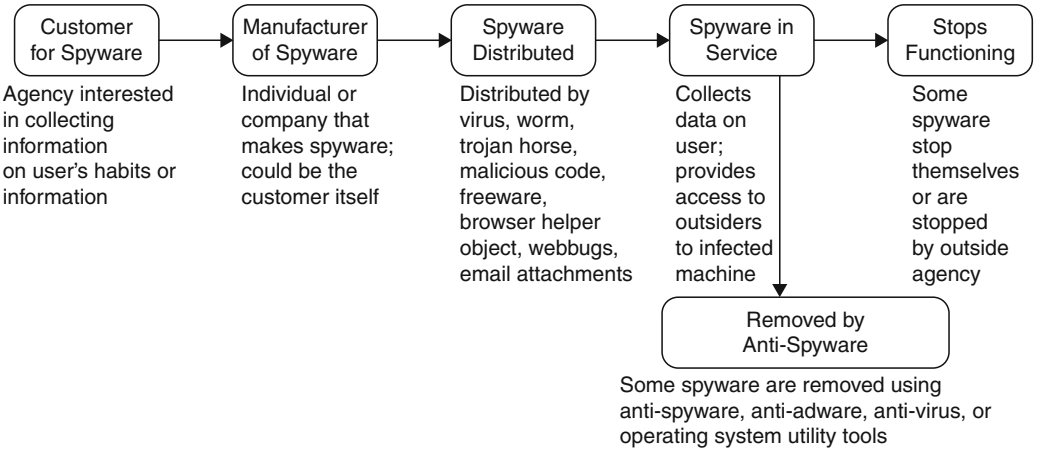
When a person goes to an on-line bookseller's website (such as Amazon.com), logs in with her id and password, then selects a book to purchase, provides credit card details, and then purchases the book – the person is confident of the transaction being secure since Amazon.com is a trusted site and uses secure technologies for collecting information; however, the person may not realize that there could be an eavesdropper in her own machine that has monitored all her key strokes and stored them, and at a convenient time transmits them to an external agent. Now the agent has all the keystrokes the person made, can parse the different fields (such as www.amazon.com, user id, password, etc.), and get personal data that the person would normally not want others to know. That "eavesdropper" is a software program called keylogger whose job is to record all keystrokes made by a user, and keylogger is an example of spyware. But the worst part is that many people infected with a keylogger do not realize that the spyware exists. Other kinds of spyware include email redirector that redirects all incoming and outgoing emails to another one for monitoring [5], screen recorder that regularly takes snapshots of screens and sends them to an external agency [5], chat loggers, URL recorders, and adware that pops-up ads based on user preferences or the sites the user visits.

One of the most noticeable impact of the presence of spyware is the often significant reduction in the performance of the computing device since memory and processing cycles are consumed by the spyware running in the background. So how does the spyware infect a user's computing device? Most of the time spyware gets installed by the actions of the users themselves such as by clicking a link on an email which may download a spyware program or by subscribing to free offers from a website that in turn downloads the spyware as well, or it may be supplied as a trojan wherein it is seemingly benign in application but is actually a spyware. Spyware has also been found bundled as part of procured software – both free and purchased. If

the spyware is an adware, then the users will immediately notice a spurt in ads popping up on the computing device that was not occurring earlier; adware also tends to overwrite hosts file on computing devices that directs users to advertisers'(or other interested parties') sites instead of the usual homepage. If the spyware is more insidious such as a keylogger, then other than a performance hit of the device the users may not notice any explicit behavior change of the device. There is a class of spyware called "ransomware" that displays frightening scenarios of infection in a user's device, the removal of which requires private information to be provided by the user; a notorious example of this type is the Antivirus Soft [1] that indicates usually misleadingly that users have been infected with different types of malware, and that if they purchase Antivirus Soft, all problems can be fixed – this spyware is actually masquerading as an anti-spyware software! Rootkit is another type of spyware that hides its presence from other applications (including anti-spyware programs) by manipulating the operating system but at the same time getting access to users' information from the system.

It is difficult to get rid of spyware from a computing device and that can happen only when users realize that they have been infected in the first place! Typically, anti-spyware programs can help detect and isolate most spyware – however, they need to be kept updated by the users so that information on new spyware is added to the existing anti-spyware software. Some anti-spyware programs also prevent spyware infections. Otherwise the only solution may be to format the device's mass storage (the hard disk for computers) and re-image the operating system and applications. Symantec's Norton Internet Security software is an example of anti-spyware protection program; others include Trend Micro's antivirus and anti-spyware, McAfee Anti-Spyware, and Microsoft's Windows Defender. There are both single-user versions of anti-spyware as well as enterprise versions that take care of the needs at the enterprise level – Symantec's Endpoint Protection software is an anti-spyware software program that blocks spyware infection at the enterprise level. For Rootkits, there are several detectors including Microsoft's Sysinternals Rootkit Revealer, Sophos' Anti-Rootkit, and F-Secure's Blacklight.

The lifecycle of spyware is given in Fig. 1. Firstly, there must be an interested customer for the spyware such as an advertising company, employer, user habits tracking companies, or government agencies. The customer contacts a programmer or a spyware manufacturing company to develop the spyware for the customer's requirements; sometimes the customer may be the spyware manufacturing company itself. The spyware is then distributed by the customer or the manufacturer using one of several techniques: bundled with freeware, bundled with purchased software, as an email attachment, a trojan, a browser helper object, a webbug, or even as a virus or worm. Typically, a spyware attaches to the user's computer by a user action such as clicking a link or an attachment. Once installed on the user's machine, the spyware monitors the user's computer usage habits and transmits this information to the customer of the spyware through a back door over the internet created on the user' device; the customer profits by selling this information or by the knowledge obtained. Usually spyware continues to operate in the background without the user ever being aware of its presence. Sometimes spyware gets rid of itself after a period of time or when the customer shuts it down through the back door. Sometimes spyware is removed by anti-spyware, anti-adware, or antivirus applications, or by operating system utility tools. The life cycle of a spyware is usually



Spyware. Fig. 1 Lifecycle of spyware

independent of the operating system used on the computing device.

A recent phenomenon has been the emergence of spyware in mobile computing devices [6] due to the fact that several of these devices can access the web; however, not many anti-spyware software are available for mobile platforms. Currently, mobile devices usually rely on the features of their operating systems to protect them; sometimes, administrators regularly "flash" their companies' fleet of mobile devices in which they erase and reprogram the devices, and this will also help remove spyware.

Applications

The main reason for developing spyware is to spy user's preferences and habits without their knowledge. While spyware generally has a negative connotation due to the lost hours in productivity they cause to the users, there are some beneficial purposes as well. For example, it is possible for employers to scrutinize more thoroughly the computing device user habits of their employees – using spyware (such as a keylogger) it will be possible to examine exactly what programs, sites, and contents the employees used, visited, or entered; likewise, schools can view the browsing habits of their students; and parents can more accurately keep in touch with their wards' computing habits. It will also be possible to monitor exactly how many hours an employee has worked on her computer for productive purposes from the viewpoint of the business or for schools to monitor how efficiently students used their computers. However, privacy issues will need to be considered for these more benign uses of spyware.

Open Problems

One of the issues involved with spyware is recognizing whether it is benign or not and ensuring benign spyware is allowed to run while the other category of spyware is eliminated – this is not easily done with anti-spyware software without professional help for most users. Spyware programs have usually kept ahead of anti-spyware technology, and many anti-spyware software are not very effective in spyware removal – means of reversing this trend need to be explored: adaptive anti-spyware programs may be one approach. Another issue is that both spyware and anti-spyware software affect the performance of the host computing device by using CPU cycles for execution – it will be better if hardware-based anti-spyware protection could be found since they are usually faster than software-based solutions. An important issue with spyware and its antidote is how to deal with them in mobile devices since not many resources can be spared for both of these software in these devices.

Recommended Reading

1. Spy-proofing your Computer, Shiela Ward (2005) http://www.topsecretsoftware.com/spy-proofing.html. Accessed 26 Aug 2010
2. Kirk J (2008) ComputerWorld.com. Analyst: money will lead to more mobile spying programs. http://www.computerworld.com/s/article/9072798/Analyst_ Money _will_lead_to_more_mobile_spying_programs. Accessed 30 Aug 2010
3. Evans GD (2005) Spyware reference and study guide, 1st edn. LIGATT Publishing, Beverly Hills
4. Remove antivirus soft (uninstall guide) (2010) http://www.bleepingcomputer.com/virus-removal/remove-antivirus-soft. Accessed 27 Aug 2010
5. Consumer report (2006) "State of Spyware Q2 2006". http://www.webroot.com/resources/stateofspyware/excerpt.html. Accessed 5 Oct 2010
6. Federal Trade Commission (2008) Consumer fraud and identity theft complaint data January–December 2007. www.ftc.gov/opa/2008/02/fraud.pdf. Accessed 5 Oct 2010
7. Guide: virus, spyware, and malware removal for windows (2008) http://www.winvistaclub.com/s5.html. Accessed 14 Oct 2010

SQL Access Control Model

Sabrina De Capitani di Vimercati, Giovanni Livraga
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Grant Option; ►Privileges in SQL; ►Recursive Revoke

Definition

The SQL access control model defines which authorization identifiers (i.e., users) can access specific data.

Theory and Applications

SQL access control is based on ►*privileges* assigned to *authorization identifiers* to access *objects* [1–3]. The creator of an object in a database is its owner and can perform any action on the object. By default, no other user can access the object unless the owner grants specific privileges to that user. The granting process assigns a privilege on an object to one or more *authorization identifiers*, which can be either *user identifiers*, *role names*, or `PUBLIC`. A user identifier represents a user of the DBMS and is defined in an implementation-dependent way; SQL does not define how OS users are mapped to SQL users. A role name *R* represents a role and identifies a set of privileges: those directly granted to *R* and those of the roles granted to *R*. The `PUBLIC` identifier is used to assign privileges to all the authorization identifiers in the SQL environment.

When an SQL-session is initiated, the subject that starts it has associated a pair of identifiers ⟨*uid*, *rid*⟩, where *uid* is the SQL-session user identifier, which can never be null, and *rid* is a role name whose value is initially null. Since during an SQL-session the pair ⟨*uid*, *rid*⟩ can change, a sequence of pairs ⟨*uid*, *rid*⟩ is stored within an *authorization stack* maintained using a "last-in, first-out" discipline. At a given time, the current pair ⟨*uid*, *rid*⟩ (i.e., the pair on top of the authorization stack) determines the privileges for the execution of each SQL statement. Note that *uid* and *rid* cannot both be null, that is, there must always be a non-null current *uid* or current *rid* to have privileges to execute some SQL statements.

Roles are created with a `CREATE` statement that includes the name <role name> of the role and may specify the grantor by means of the "`WITH ADMIN` <grantor>" clause. In particular, if that clause is not specified, the grantor is set to the non-null value in the current pair ⟨*uid*, *rid*⟩. Otherwise, the <grantor> can be `CURRENT_USER` or `CURRENT_ROLE`. For instance, suppose that the current *uid* is `Alice` who executes the SQL statement "`CREATE` Supervisor." As a result, role Supervisor is created. Authorization identifiers can activate a role if the role has been granted to them by means of a `GRANT` statement. A `GRANT` statement includes one or more names (<role granted>) of the roles granted, one or more authorization identifiers (<grantee>) of the grantees, and the optional clauses "`WITH ADMIN OPTION`" and "`GRANTED BY` <grantor>." `WITH ADMIN OPTION` specifies whether the grantee may grant the role to others and `GRANTED BY` specifies the grantor. Note that no cycles of role grants are allowed. Whenever an authorization identifier creates or grants a role, a *role authorization descriptor* is created in the SQL environment. A role authorization descriptor includes the role name, the authorization identifier of the grantor, the authorization identifier of the grantee, and an indication of whether the role is grantable. Role authorization descriptors can be revoked by means of the `REVOKE` statement, which removes all the identified role authorization descriptors. Revocation can be requested with or without *cascade*: in a cascading `REVOKE`, the identified role authorization descriptor is revoked together with all role authorization descriptors based on the revoked role.

An SQL privilege granted on an object to a specified authorization identifier authorizes such identifier to access the object according to the type of privileges defined (►Privileges in SQL). Privileges are managed through the `GRANT` and `REVOKE` statements. The `GRANT` statement grants a privilege on an object to one or more authorization identifiers. Figure 1 illustrates the syntax for the `GRANT` statement. `ALL PRIVILEGES` is equivalent to the specification of all the privileges on <object name> for which the grantor has grantable privilege descriptors. If the `WITH HIERARCHY OPTION` clause is present, the action must be `SELECT` and the object must be a table. This clause provides the grantee with `SELECT` privileges on all subtables (both existing and any that may be added in the future) of the table on which the privilege is granted. When the `WITH GRANT OPTION` clause is present in the statement, the users receiving the privileges have the additional authority to extend these privileges to other users (►Grant option). The "`GRANTED BY` <grantor>" clause can coincide with "`GRANTED BY CURRENT_USER`" or "`GRANTED BY CURRENT_ROLE`," meaning that the grantor specified in the privilege descriptor must be either the current user identifier or the current role name, respectively. If this clause is not specified, the grantor is set to the current user identifier, if it is not null; role name, otherwise.

Grants are represented by privilege descriptors that can be graphically represented as a graph called *privilege dependency graph*. The privilege dependency graph is a directed graph such that each node represents a privilege descriptor and each arc from node p_1 to node p_2 represents the fact that p_2 directly depends on p_1. The privilege descriptor p_2 directly depends on the privilege descriptor p_1 if the following conditions hold: (1) the privilege represented by p_1 is grantable; (2) p_1 and p_2 have the same action and object; (3) the grantee of p_1 is either the same as the grantor of p_2 or is `PUBLIC`, or, if the grantor of p_2 is a role name, the grantee of p_1 belongs to the set of applicable roles of the grantor of p_2. As an example, suppose that *Alice* is the owner of table `Employee`, and that a possible sequence of grant statements is the following (the grantor is reported in brackets): (1) (*Alice*) `GRANT SELECT ON TABLE Employee TO` *Bob* `WITH GRANT OPTION`; (2) (*Bob*) `GRANT SELECT ON TABLE Employee TO` *Carol* `WITH GRANT OPTION`; (3) (*Bob*) `GRANT SELECT ON TABLE Employee TO` *David*; (4) (*Carol*) `GRANT SELECT ON TABLE Employee TO` *Emma*. Figure 2a illustrates the privilege dependency graph for this sequence of `SELECT` privilege grants, where nodes are labeled with the grantor/grantee identifier.

An authorization identifier can then execute all actions allowed by the set of applicable privileges for it. In particular, the set of user privileges for a user identifier consists of all privileges defined by the privilege descriptors whose grantee is either that user identifier or `PUBLIC`. Analogously, the set of role privileges for a role name consists of all privileges defined by the privilege descriptors whose grantee is either that role name, `PUBLIC`, or one of the applicable roles of that role name.

The REVOKE statement revokes privileges from authorization identifiers that have been previously granted. Figure 1 illustrates the SQL syntax for the REVOKE statement. A REVOKE statement identifies the set of all privilege descriptors where the action and object are equal to that explicitly or implicitly specified in <action> and <object name>, respectively, the grantor is <grantor>, if specified, or coincides with the non-null element in the current pair $\langle uid, rid \rangle$, and the grantee is <grantee>. Privileges revocation can be performed with *cascade* or *restrict* options. Cascading revoke (►Recursive revoke) deletes privileges that recursively depend on the privilege explicitly revoked. As an example, consider the revoke statement: REVOKE SELECT ON Employee FROM *Carol* CASCADE, issued by *Bob*. Figure 2b illustrates the result of the execution of this statement on the graph of Fig. 2a. In particular, the effect of this REVOKE statement is to recursively remove from the privilege dependency graph nodes representing the SELECT privilege on table Employee granted to *Carol* and *Emma*. If the REVOKE statement includes the RESTRICT option, the privilege will not be revoked if there are any dependent privileges. For instance, the REVOKE statement REVOKE SELECT ON Employee FROM *Carol* RESTRICT issued by *Bob* would be rejected since there is a dependent privilege (i.e., *Emma*'s privilege).

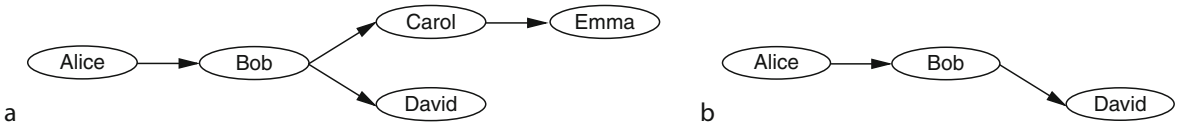
Access Control Enforcement via Views

In addition to using the privileges assignment mechanism described above to provide specific types of access to entire objects, there is also a mechanism within SQL to provide access to only certain portions of the contents of objects (i.e., to enforce content-based restrictions). This mechanism is based on the definition of *views* (View-based Access Control). A view is a virtual table derived from base tables and/or other views. A view is specified by a *view definition* that is written in SQL. The database stores the views definitions and materializes the view as needed. The general syntax of the CREATE VIEW statement includes the name of the view, the name of the structured type, if any, associated with the table or the list of its columns (<view specification>), the <query expression> that defines how the view is to be obtained, and an indication of whether the view is updatable or not (CHECK OPTION clause that can be applied as CASCADED or LOCAL). When a view is created, it is necessary to verify whether the creator has permission to access all the views or base tables directly referenced by the view. The creator of the view then receives on it the privileges that she holds on all the tables and views directly referenced; the grant option on the privileges received applies only if the creator of the view has the grant option on all the tables used in the definition of the view. In this case, the creator of the view can grant to others access to the view. This has the effect that these other users have access to the information presented by the view, which they might not otherwise be able to access. In general, views can restrict access in a number of ways: (1) to specific columns, (2) to specific rows, or (3) to aggregates of the information accessed by the view, using the *aggregate functions* supported by

```
GRANT ALL PRIVILEGES | <action>
ON [ TABLE ] | DOMAIN | COLLATION | CHARACTER SET | TRANSLATION | TYPE <object name>
TO <grantee> [{<comma> <grantee>}...]
[ WITH HIERARCHY OPTION ]
[ WITH GRANT OPTION ]
[ GRANTED BY<grantor>]
```

```
REVOKE [ GRANT OPTION FOR | HIERARCHY OPTION FOR ]<action>
ON [ TABLE ] | DOMAIN | COLLATION | CHARACTER SET | TRANSLATION | TYPE <object name>
FROM <grantee>[ {<comma> <grantee>}... ]
[ GRANTED BY <grantor> ]
CASCADE | RESTRICT
```

SQL Access Control Model. Fig. 1 Syntax of the SQL statements for managing privileges



SQL Access Control Model. Fig. 2 An example of privilege dependency graph (**a**) the same graph after a REVOKE statement (**b**)

SQL (e.g., SUM, MIN, MAX). When updates are considered, views and base tables must be treated differently. In general, users cannot directly update views, particularly when they are created from the combination (join) of two or more base tables. The updatable views can instead appear as targets of statements that change SQL data. The results of changes expressed in this way are defined in terms of corresponding changes to base tables.

Recommended Reading

1. De Capitani di Vimercati S, Samarati P, Jajodia S (2001) Database security. In:Marciniak J (ed) Wiley Encyclopedia of Software Engineering. Wiley, New York
2. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In:Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. LNCS, vol 2171. Springer, Berlin
3. Database Language SQL (2008) ISO International Standard, ISO/IEC 9075–*:2008

SQL Injection Attacks

Alessandro Orso
College of Computing - School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Definition

Structured Query Language injection attacks (SQLIAs) are one of the major security threats for Web applications [1]. SQLIAs are a class of code injection attacks that take advantage of a lack of validation of user input. These attacks occur when developers combine hard-coded strings with user-provided input to create dynamic queries to an underlying database. Intuitively, if user input is not properly validated, attackers may be able to change the developer's intended Structured Query Language (SQL) command by inserting new SQL keywords or operators through specially crafted input strings. As a result, attackers can get access to (and even control) the underlying database, which may contain sensitive or confidential information. Despite the potential severity of SQLIAs, many Web applications remain vulnerable to such attacks.

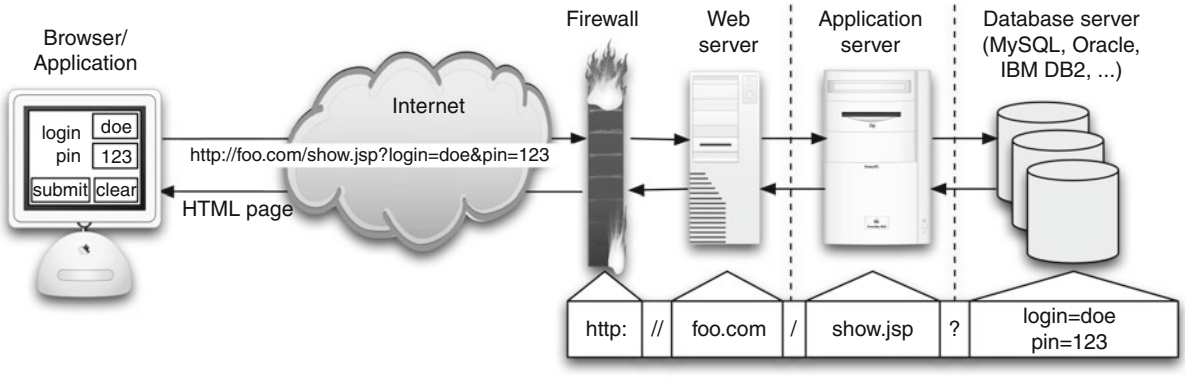
Background

SQL is a language for querying and updating the information contained in relational databases. *Web applications* are multi-tier, distributed applications where the front end is typically a Web browser and the back end is a server that relies on one or more underlying databases (and possibly on other servers).

Theory

Figure 1 shows an example of a typical Web-application architecture. In the example, the user interacts with a Web form that takes a login name and pin as inputs and submits them to a Web server. The Web server passes the user-supplied credentials to a *servlet* (`show.jsp`), which is a special type of Java application that runs on a Web application server, and whose execution is triggered by the submission of a URL from a client. The servlet queries the database and generates a response for the user in the form of a Web page.

The example servlet, whose code is partially shown in Fig. 2, implements a login functionality that can be found in a typical Web application. It uses input parameters `login` and `pin` to dynamically build an SQL query or command. (Hereafter, for simplicity, the terms *query* and *command* are used interchangeably.) The `login` and `pin` are checked against the credentials stored in the database.



SQL Injection Attacks. Fig. 1 Example of interaction between a user and a typical Web application

```
1. String login = getParameter("login");
2. String pin = getParameter("pin");
3. Statement stmt = connection.createStatement();
4. String query = "SELECT acct FROM users
   WHERE login='";
5. query += login + "' AND pin=" + pin;
6. ResultSet result = stmt.executeQuery(query);
7. if (result ! = null)
8.     displayAccount(result); //Show account
9. else
10.    sendAuthFailed(); // Authentication failed
```

SQL Injection Attacks. Fig. 2 Excerpt of a Java servlet implementation

If they match, the corresponding user's account information is returned. Otherwise, a null set is returned by the database and the authentication fails. The servlet then uses the response from the database to generate HTML pages that are sent back to the user's browser by the the Web server.

For this servlet, if a user submits `login` and `pin` as "`doe`" and "`123`," the application dynamically builds the query:

```
SELECT acct FROM users WHERE login='doe' AND pin=123
```

If `login` and `pin` match the corresponding entry in the database, `doe`'s account information is returned and then displayed by function `displayAccount()`. If there is no match in the database, function `sendAuthFailed()` displays an appropriate error message. An application that uses this servlet is vulnerable to SQLIAs. For example, if an attacker enters "`admin' --` " as the user name and any value as the pin (e.g., "`0`"), the resulting query is:

```
SELECT acct FROM users WHERE login='admin' -- ' AND pin=0
```

In SQL, "`--`" is the comment operator, and everything after it is ignored. Therefore, when performing this query, the database simply searches for an entry where `login` is equal to `admin` and returns that database record. After the "successful" login, the function `displayAccount()` reveals the `admin`'s account information to the attacker.

It is important to stress that this example represents an extremely simple kind of attack that is presented for illustrative purposes only. Because simple attacks of this kind are widely used in the literature as examples, they are often mistakenly viewed as the only types of SQLIAs. In reality, there is a wide variety of complex and sophisticated SQL exploits available to attackers, including:

- Tautologies
- Union queries
- Piggy-backed queries
- Malformed queries
- Inference
- Alternate encodings
- Leveraging stored procedures

See Reference [2] for a formal definition of SQLIA and Reference [3] for a detailed discussion of the different types of attacks listed above.

Open Problems

Although it is still possible to find many Web applications that are vulnerable to SQLIAs, the problem of SQL injection per se can be considered mostly solved. In fact, in the last few years, researchers have defined a number of effective techniques for detecting and preventing SQLIAs. The presented techniques include approaches based on static analysis, on dynamic analysis, on combination of the two, and on the use of specialized languages that are not vulnerable to injections. (Reference [3] discusses the most relevant approaches presented in the literature to date.)

Recommended Reading

1. The OWASP Foundation (2007) Top ten most critical web application vulnerabilities. http://www.owasp.org/images/e/e8/OWASPTop102007.pdf
2. Su Z, Wassermann G (2006) The essence of command injection attacks in web applications. In: Proceedings of the annual symposium on principles of programming languages, Charleston, pp 372–382
3. Halfond W, Orso A, Manolios P (2008) WASP: protecting web applications using positive tainting and syntax-aware evaluation. IEEE Transactions on Software Engineering 34(1):65–81

Square-and-Multiply Exponentiation

►Binary Exponentiation

SSH

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Secure shell

Definition

SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices.

Background

Used primarily on Linux- and Unix-based systems to access shell accounts, SSH was designed as a replacement for Telnet and other insecure remote shells, which send information, notably passwords, in plaintext, rendering them susceptible to packet analysis. For SSH2, there was in addition a replacement for FTP, namely sftp.

For some time, it was developed as a standard under IETF. When the standardization was terminated, there were two versions of Secure Shell available – SSH1 and SSH2 – which unfortunately are quite different and incompatible.

Theory

SSH basically provides strong ►*authentication* and secure communications (confidentiality and integrity of data) over insecure channels such as the Internet.

As for the use of cryptographic algorithms, SSH1 supported DES (the ►*Data Encryption Standard*) ►*Triple-DES, IDEA*, and ►*Blowfish*, for encryption, while SSH2 supports 3DES, Blowfish, ►*Twofish*, and a few others. For authentication, SSH1 supported ►*RSA digital signature scheme*, while SSH2 supports the Digital Signature Standard.

Applications

SSH allows a user to log into another machine over a network, to execute commands in a remote machine, and to move files from one machine to another.

Recommended Reading

1. Barrett DJ, Silverman RE, Byrnes RG (May 2005) SSH, The secure shell: the definitive guide. O'Reilly, Sebastopol
2. www.openssh.com

SSL

►Secure Socket Layer (SSL)

SSS

►Secret Sharing Schemes

Stack (Buffer) Overflow

►Buffer Overflow Attacks

Stack (Buffer) Overrun

►Buffer Overflow Attacks

Stack/Heap Smashing

►Buffer Overflow Attacks

Standard Basis

►Gröbner Basis

Standard Model

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Generic Model; ►Random Oracle Model; ►Security Proofs

Definition

In cryptography, the standard model is the model of computation in which the adversary is only limited by the amount of time and computational power available. Cryptographic constructions are usually based on complexity assumptions, which state that some problem, e.g., factorization or the discrete logarithm problem, cannot be solved in polynomial time. Schemes which can be proven secure using only complexity assumptions are said to be secure in the standard model. While several security proofs in the standard model exist, security proofs are notoriously difficult to achieve in the standard model.

Open Problems

Find a stateless RSA based signature scheme secure in the standard model.

Recommended Reading

1. Rogaway P (2004) On the role definitions in and beyond cryptography. In: Maher MJ (ed) ASIAN 2004. LNCS, vol 3321. Springer, Heidelberg, pp 13–32

Static Analysis

David Brumley
Electrical and Computer Engineering Department, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms

Static program analysis

Related Concepts

►Dynamic Analysis; ►Static Analysis

Definition

Static analysis is the process of analyzing a program text in order to extract semantic information.

Background

Static analysis algorithms historically have come from compiler research and implementations. Static analysis algorithms extracted facts about a program that were used by the compiler to ensure correct and efficient translation from a source language to a destination language. One of the earliest examples is the 1957 IBM FORTRAN compiler, which included a number of static analyses for efficiently mapping user variables to registers, transforming loops to make execution more efficient, and ensuring a program is well typed [1].

Initial static analysis were *intra-procedural* analysis, meaning only a single procedure is analyzed at a time, and all procedures are analyzed independent of other procedures. *Inter-procedural* analysis — static analysis that considers the relationships between procedure calls — is more expensive in terms of memory and space, thus did not appear in early compilers. Modern compilers make use of both intra-procedural and inter-procedural analysis.

Static analysis algorithms that are specifically designed to check security, safety, and portability properties have grown organically out of compiler research. Standard compilers textbooks, such as *Compilers: Principles, Techniques, and Tools* [2] (aka the "Dragon" book), are considered standard references within the software security community. Stand-alone tools for checking such properties date back to at least the 1970s, e.g., the UNIX V7 "lint" tool checks for program constructs that are likely to be bugs. Currently there are both freely available security-specific static analysis tools such as RATS (Rough Auditing Tool for Security, [3]) and ITS4 [4], as well as commercial tools such as Coverity [5] and Fortify [6].

Theory

Static analysis analyzes the program text in order to extract facts about the program. Finding an exact solution to typical static analysis questions is almost always undecidable. This fact follows from Rice's theorem, as most questions in static analysis will be nontrivial program properties [7, 8]. Static analysis is often contrasted with dynamic analysis. Unlike dynamic analysis, which analyzes a single execution, static analysis can analyze all available program text, and is not restricted to a single program path.

Due to the undecidable nature of static analysis properties, static analysis algorithms typically strive for approximate solutions. There are two primary types of approximations. First, an approximation may be sound, meaning whatever the analysis says must, in fact, be true in the real program. Second, an approximation may be complete, meaning it says all true facts about the real program. Note that an algorithm that is both sound and complete is exact, thus typically not possible: a choice must be made. Somewhat more rare are algorithms that make no guarantees.

The overall accuracy of a static analysis algorithm is usually understood by how precisely the algorithm models program paths, both intra-procedurally and inter-procedurally. In a program with loops, there are an infinite number of possible paths. A possible path is a path based upon the syntax of the program, and in particular, does not take into consideration whether a particular branch may ever actually execute. For example, an analysis may accurately distinguish between paths in a single procedure, but not individual paths over function calls (i.e., called path-sensitive but calling context insensitive typically in literature).

There are many approaches to static analysis, including:

Data flow analysis Data flow analysis is a static analysis technique for calculating facts of interest at each program point based upon the control flow graph representation of the program. Gary Kildall formalized data flow analysis in 1973 in terms of lattice theory [9]. Facts about the program are vertices in the lattice. The lattice meet operator determines how to combine two sets of facts, e.g., at the confluence of two branches in a program.

There are dozens of typical data flow analyses that can be found in common compiler textbooks [2]. For example, a canonical data flow analysis is reaching definitions. If statement i is $x := y \square z$, where $\square$ is any operator, that x is defined at statement i. Reaching definitions determines for each statement j that uses a variable x which previous statement defined x. In

```
1 x := y + z;
2 w := x + z;
3 x := w;
```

The definition x at line 1 reaches line 2, but does not reach after line 3 because x is assigned on line 3. A typical application of data flow analysis to security is to determine whether a particular program variable is derived from user input (tainted) or not (untainted). Given an initial set of variables initialized by user input (i.e., initialized as tainted in the lattice), a simple data flow analysis can calculate (typically an over approximation) of all variables in the program that are derived from user data.

Given an analysis where the lattice meet operator is well-defined and the lattice is of finite height, a data flow analysis is guaranteed to terminate and converge to an answer for each statement in the program using a simple iterative algorithm.

Abstract interpretation Abstract interpretation [10], first formalized by Patrick and Radhia Cousot in 1977 [11], executes a program on an abstract machine to determine the properties of interest. The level of detail in the abstract machine specification typically determines the accuracy of the results. At one end of the spectrum is an abstract machine that faithfully represents the concrete semantics of the language. Abstract interpretation on such a machine would correspond to a concrete execution on the real machine. Typical abstract interpretation analyses, however, typically do not require detailed semantics. For example, an abstract interpretation that determines the sign of each addition and multiplication operation, assuming no overflow, is fairly simply where each variable has associated with it the value $\{+, -, 0\}$.

Model checking Model checking is a method for formally verifying that a model satisfies a specified property. Model checking algorithms typically entail enumerating the program state space to determine if the desired properties hold [12]. The MOPS (MOdelchecking Programs for Security Properties, [13]) and CBMC (C Bounded Model Checker, [14]) are example model checkers that can be used to check security properties of programs.

Type checking Type checking is a form of static analysis which checks that program statements are well-formed with respect to a typing logic [15]. For example, integers can be added, functions can be called, but functions should not be added and integers should not be called. Type checking can be used to ensure programs are type-safe, meaning that at every step of the execution all values have well-defined and appropriate types, and that there is a valid next step of execution.

Applications

Typical static analysis questions in security include how large a buffer may be, whether an integer operation will result in overflow, and what constraints does the program impose in order to execute a specific line of code. Determining how large a buffer may be is important to determine whether an operation is vulnerable to a buffer overflow attack. Determining whether an integer operation will overflow is important for detecting integer overflow vulnerabilities. Determining constraints to execute a program statement are important for determining whether a potentially vulnerable line of code can actually be exploited.

Static analysis algorithms can also aide in securing a program by inserting and optimizing run-time safety checks. For example, a static analysis may insert and optimize checks that make sure copies into buffers never exceed the buffer size in an unsafe language such as C. Another example is optimizing away checks, e.g., in Java a bounds check is inserted at every array dereference to ensure safety. Static analysis then can remove unneeded bounds checks, e.g., a check on an array operation that has already been determined to be within bounds.

Recommended Reading

1. Backus JW, Beeber RJ, Best S, Goldberg R, Haibt LM, Herrick HL, Nelson RA, Sayre D, Sheridan PB, Stern H, Ziller I, Hughes RA, Nutt R (1957) The FORTRAN automatic coding system. In: Proceedings of the Western Joint Computer Conference: papers presented at the Joint IRE-AIEE-ACM Computer Conference, Los Angeles, 26–28 February 1957. Institute of Radio Engineers, New York
2. Aho AV, Lam M, Sethi R, Ullman JD (2007) Compilers: principles, techniques, and tools, 2nd edn. Addison-Wesley, Boston
3. http://www.fortify.com/security-resources/rats.jsp
4. http://www.cigital.com/its4
5. http://www.coverity.com
6. http://wwww.fortify.com
7. Hopcroft JE, Motwani R, Ullman JD (2001) Introduction to automata theory, langauges, and computation. Addison-Wesley, Boston
8. Landi W (1992) Undecidability of static analysis. ACM Lett Progr Lang Sys 1(4):327–337
9. Kildall GA (1973) A unified approach to global program optimization. In: POPL '73: Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Boston, 1–3 October 1973. ACM Press, New York, pp 194–206
10. Nielson F, Nielson HR, Hankin C (2004) Principles of program analysis. Springer, Berlin

11. Cousot P, Cousot R (1977) Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference record of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles, 17–19 Jan 1977. pp 238–252
12. Clarke EM, Grumberg O, Peled DA (1999) Model checking. MIT Press, Cambridge, MA
13. http://www.cs.berkeley.edu/~daw/mops/
14. http://www.cs.cmu.edu/~modelcheck/cbmc/
15. Pierce BC (2002) Types and programming languages. MIT Press, Cambridge, MA

Static Code Analysis

►Program Verification and Security

Static Program Analysis

►Static Analysis

Static Separation of Duties

►Separation of Duties

Station-to-Station Protocol

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Definition

The Station-to-Station protocol is a popular authenticated key exchange.

Theory

In a two-party *authenticated key exchange*, the legitimate parties can compute a secret key, while at the same time being certain about the authenticity of the parties with whom they exchange a key. The scheme must, in particular, be secure against a ►man-in-the-middle attack.

A popular authenticated version of the ►Diffie–Hellman key exchange protocol is the ►Station-to-Station protocol. It was proposed by Diffie–van Oorschot–Wiener [1].

Let $\langle g \rangle$ be a suitable finite cyclic group of large enough order in which the ►computational Diffie–Hellman problem is (assumed to be) hard. We assume that q (not necessarily prime) is a multiple of the order of g and is publicly known. Let $sign_A(m)$ indicate the digital signature of the bitstring m by party A. So, $sign_A(m)$ can be verified using the public key of A. Let $E_k(m)$ be a conventional encryption of the bitstring m using the conventional key k. (If k is too long, one assumes it is hashed). The corresponding decryption is written as $D_k(\cdot)$.

The protocol in which Alice (A) and Bob (B) want to exchange a key works as follows:

Step 1 A sends B $\alpha := g^{r_A}$ computed in $\langle g \rangle$, where r_A is chosen uniformly random in Z_q.

Step 2 B chooses r_B uniformly random in Z_q and computes $\beta := g^{r_B}$ in $\langle g \rangle$, $k_B := \alpha^{r_B}$ in $\langle g \rangle$ and $\gamma_B := E_{k_B}(sign_B(\alpha, \beta))$, where α and β are concatenated. B sends A: β, γ_B.

Step 3 A computes $k_A := \beta^{r_A}$ and verifies whether the string $D_{k_A}(\gamma_B)$ is the digital signature of (α, β), signed by B. If so, she sends B: $\gamma_A := E_{k_A}(sign_A(\alpha, \beta))$ and views k_A as the authenticated key exchanged with B.

Step 4 B verifies whether the string $D_{k_B}(\gamma_A)$ is the digital signature of (α, β) signed by A. If so, B regards k_B as the authenticated key exchanged with A.

As in the Diffie–Hellman scheme, if there are no dishonest parties, Alice and Bob will exchange the same key, i.e., $k_A = k_B$.

Recommended Reading

1. Diffie W, van Oorschot PC, Wiener MJ (1992) Authentication and authenticated key exchanges. Designs, Codes Cryptogr 2:107–125

Statistical Databases

Nabil Adam[1], Haibing Lu[1], Jaideep Vaidya[1], Basit Shafiq[2]
[1]Department of Management Science and Information Systems, CIMIC, Rutgers, The State University of New Jersey, Newark, NJ, USA
[2]CIMIC, Rutgers, The State University of New Jersey, Newark, NJ, USA

Synonyms

Multidimensional databases; Online analytical processing

Related Concepts

►ε-Privacy; ►Data Cube; ►Inference Control

Definition

A statistical database (SDB) system is a database system that enables its users to retrieve only aggregate statistics (e.g., sample mean and count) for a subset of the entities represented in the database.

Background

As a statistical database may contain sensitive individual information, such as salary and health records, generally, users are only allowed to retrieve aggregate statistics for a subset of the entities represented in the databases. Common aggregate query operators in SQL include SUM, COUNT, MAX, MIN, and AVERAGE, though more sophisticated statistical measures may also be supported by some database systems.

Statistical databases pose unique security concerns, which have been the focus of much research. However, the key security challenge is that of ensuring that no user is able to infer private information with respect to a privacy disclosure policy while providing as much statistical information, as possible. Even though a single aggregate query may not release much information on individual records, responses to a series of aggregate queries could enable an adversary to infer sensitive information. The following simple example demonstrates this: Consider a university payroll database that contains salary information about all faculty and staff. Supposing a user would like to know the salary of Prof. X, who he knows is the only female professor in the Mechanical Engineering Department. Now, the user submits the first query asking for the sum of the salaries of all the professors in the Mechanical Engineering Department, and the second query asking for the sum of the salaries of all male professors in the same department. While both queries are individually innocuous, when the result of the second query is subtracted from that of the first query, the salary of the sole female professor in the department is revealed.

Although statistical databases have been studied for over 40 years, there is no universal solution for securing statistical databases in all situations. The many challenges are as follows: (1) It is difficult to determine the a priori knowledge of a malicious user; (2) users may collude; (3) it is difficult to develop a universal solution for statistical databases of various types, such as Offline versus Online, Static versus Dynamic, and Centralized versus Decentralized; (4) it is always a computationally challenging problem to provide a good trade-off between data privacy and data utility.

Applications

Statistical databases have been applied in many areas, including manufacturing test, health care analysis, and business reporting. A typical example of a statistical database is the database maintained by the US Census Bureau, since its primary purpose is to provide aggregate statistics on the population.

Privacy Disclosure Measures

Privacy disclosure policies for statistical databases can be roughly classified into exact disclosure and partial disclosure. With exact disclosure, the privacy of an individual record is considered to be breached if its exact value is revealed. On the other hand, with partial disclosure (which is more strict), privacy is considered to be compromised as soon as a certain amount of partial information about a record is disclosed. Partial disclosure policies include interval-based disclosure [1] and probabilistic disclosure [2]. In interval-based disclosure a numeric record is disclosed if its real value is determined to fall within any interval of length smaller than its predefined safe threshold. For example, a person's salary of \$150,000 is considered to be revealed if one can determine that it falls in any subinterval of (\$120,000, \$160,000) given that the safe threshold value is greater than \$40K. With probabilistic disclosure, an element is considered to be revealed if one can ensure that its real value falls in a small interval with high probability. Probabilistic disclosure is considered more strict than interval-based disclosure. This can be easily seen by the following example. Assume that from the past query answers, the age of a patient having some disease is inferred to fall within the interval of (10, 20). If the safe threshold for interval disclosure is 5, the age can still be considered private with respect to interval disclosure. However, assume that the adversary also has the prior knowledge that 95% patients having that disease are greater than 18, the adversary can improve his/her estimate of the patient's age to (18, 20) with high probability, in effect breaching probabilistic disclosure.

More recently, Dwork has proposed the concept of ►differential privacy [3] to provide a formal and quantifiable privacy guarantee for general databases. As opposed to the above two disclosure policies, differential privacy is actually a condition on the data release mechanism and not on the dataset.

Security Protection Approaches

Several security protection approaches have been proposed in the extensive literature on this topic. Corresponding to the classification scheme of Adam and Wortmann

[4], all of the approaches can be classified into conceptual, query restriction, auditing, and perturbation approaches.

Conceptual Approach

In the conceptual approach, each proposed model provides a framework for investigating the security problem at a more general data model level, without necessarily giving a specific implementation procedure. Two important conceptual approaches are the conceptual model and the lattice model. In the conceptual-modeling approach [5], security is basically guaranteed by the introduction of the concept of "smallest non-decomposable subpopulations" (also referred to as atomic populations or A-populations). These A-populations always contain either zero or at least two entities, thus preventing disclosure of information about one individual entity. Preventing A-populations from having only one entity due to insertions and deletions is controlled by either delayed update processing or by dummy entities. The idea of A-populations has been extended to the general database community to the concept of ►*k*-anonymity [6], which requires that every tuple in the database be indistinguishably related to no fewer than k tuples.

The lattice model [7] is an alternative general framework where a SDB is viewed as a high-dimensional table. As individual records contain confidential information, the table is aggregated along each dimension (corresponding to each attribute) to produce many aggregate tables. By aggregating along more dimensions, coarser aggregate tables are generated. Based on the aggregation level, all aggregate tables form a lattice. The security level can be enhanced by allowing only access to aggregate tables. Interestingly, when viewed under the lattice model, statistical databases are closely related to the data cube model employed in On line Analytical Processing (OLAP), since the data cube model in OLAP is essentially a multidimensional table. Details on similarities and differences between OLAP and SDB can be found in [8]. As such, many of the research results on addressing security problems in OLAP can be directly applied to the lattice model in SDB. However, unlike A-populations, different aggregate tables may overlap, potentially creating security vulnerabilities due to which sensitive information may be inferred through cross comparison of aggregate tables.

Query Restriction Approach

General query restriction methods include query-set-size control, query-set-overlap control, partitioning, and cell suppression. The query-set-size control method permits a statistic to be released only if the size of the query set $|C|$ satisfies the condition: $K \leq |C| \leq L - K$ where L is the size of the database and K is a parameter set by the DBA. This approach is very easy to implement, which is in favor of large-scale SDBs. However, its safety level is quite low as intelligent snoopers can compromise databases by submitting carefully designed queries. Indeed, it cannot even address the disclosure problem of the professor salary example illustrated in the introduction.

The query-set-overlap control method enhances security by restricting the number of overlapping entities among successive queries of a given user. It can be shown that the minimum number of queries needed for compromise is $1 + (K-1)/r$ where K denotes the minimum query set size and k denotes the maximum number of overlapping entities allowed between pairs of queries. However, in most cases, this is too small to meet practical needs.

The basic idea of partitioning is to cluster individual entities of the population in a number of mutually exclusive subsets, which is similar to the idea of A-populations. This is commonly done by partitioning the values of attributes. For instance, the GRADEPOINT attribute values in a STUDENT database can be divided into intervals of $([0,2],[2,3],[3,4])$. A partitioned database may still suffer from some problems. The first problem is that in reality a considerable number of atomic populations with only one entity will emerge. To fix this it may be necessary to merge distinct attributes, which leads to serious information loss. The second problem is that updates may again create atomic populations of size 1. While solutions to cope with this such as introducing dummy variables or postponing updates exist, they may introduce bias to statistics. A more serious problem is that privacy disclosure may still occur even if all entities are partitioned into mutually exclusive subsets. An example is that patient records are partitioned according to some rule such that each group contains ten records. However, one group may contain records with the HIV test results all being positive. For this case, if the average statistic of this group is revealed, test results of all records are disclosed. To cope with this, anonymization models such as ►*l*-diversity [9] have been proposed, which require the sensitive attribute values in each group to be diversified enough such that not much information about any single individual can be inferred.

Cell suppression is one of the techniques typically used by census bureaus for data published in tabular form [10]. The basic idea is to suppress from the released tables all cells that might cause confidential information to be disclosed.

Auditing

Auditing of an SDB involves keeping up-to-date logs of all queries made by each user and constantly checking

each new query for possible compromise. Auditing has advantages such as allowing the SDB to provide users with unperturbed response, provided that the response will not result in a compromise. While this used to be considered a part of query restriction, due to its unique advantages, it has received special attention. Existing auditing schemes include conventional auditing and simulatable auditing.

In conventional auditing, a query is denied when the answer to the query combined with past query answers can compromise the database with respect to the privacy disclosure policy. It is not difficult to audit queries of a single type. An efficient method of auditing SUM-only queries presented in Chin and Ozsoyoglu [11] is as follows: First, each query is represented as a 0–1 row vector with size of n, which is the number of total elements, such that the ith component is 1 if the corresponding ith element is included in the query; otherwise 0. When a new query arrives, include it with the past answered queries to form a binary matrix and perform Gaussian elimination on the matrix. If there is a row with only one component of 1 in the resulting matrix, there exists an element, the exact value of which is determined, and thus the new query should be denied. This can also be extended to interval-based disclosure. However auditing mixed SUM, MAX, and MIN queries is proven to be NP-hard due to its combinatorial nature. Indeed, it is even difficult to audit SUM-only queries with element values restricted to be binary, which is proven to be co-NP hard.

Conventional auditing also suffers from the security problem, since it ignores the important fact that query denials can also release information. Kenthapadi et al. [2] first point out this issue. A simple example can illustrate this: Assume three elements (x_1, x_2, x_3) exist, each with the value 5. The first query is $x_1 + x_2 + x_3$ and is answered as the result does not pose any threat. The second query is $max(x_1, x_2, x_3)$. If the real answer is issued, with respect to the exact disclosure policy, the security is breached as a snooper can easily infer that all elements are 5. However if the query is denied, the security is still breached, assuming that the snooper knows the conventional auditing scheme and the exact disclosure policy. The rationale is that the only feasible answer which could cause compromise, is 5.

To address this issue, Kenthapadi et al. propose the concept of simulatable auditing. The basic idea is that a query is denied if there exists even one feasible answer to the query, which is consistent with all past query answers and can cause compromise by combining it with past query answers. This is called simulatable auditing since the auditing process can be simulated by the query submitters themselves to reach the same decisions as the DBA. Simulatable auditing indeed ensures security. However, data utility is severely impaired. Consider the following example. All elements in a SDB are known to be nonnegative. If a SUM query $x_1 + x_2$ is posed. According to simulatable auditing, the query cannot be answered, because 0 is a feasible answer to the query which would determine both x_1 and x_2 to be 0. Indeed, by the same reasoning, no SUM query can be answered. More work is necessary to derive a sound auditing scheme that can derive the leaked information from each query denial and include this in the auditing process to maximize utility.

Perturbation Approach

Perturbation approaches include data-perturbation approaches and output-perturbation approaches. The former kind achieves security by modifying the source data, while the latter alters query results.

The methods based on the data-perturbation approach fall into two main categories: the probability distribution category and the fixed-data-perturbation category. The probability distribution category considers the SDB to be a sample from a given population that has a given probability distribution. In this case, the security-control method replaces the original SDB by another sample from the same distribution. In the fixed-data-perturbation category, the values of the attributes in the database, which are to be used for computing statistics, are perturbed once and for all.

Output-perturbation approaches ensure security by perturbing the query results. Approaches include random-sample queries and output perturbation. Denning [12] proposed a method that is comparable to ordinary random sampling, where a sample is drawn from the query set. For example, to answer a COUNT query, the system retrieves entries satisfying specified constraints, selects a part of retrieved entries based on some probability which is known to the user, and returns the count of the final entry set to the user. Given the selection probability, the user can estimate the real answer. This idea has been extended to the distributed environment in [13], in which aggregates need to be computed across different data agents while requiring data privacy of individual agents to be preserved during the computing process. Output-perturbation methods essentially achieve security by adding noise to query results. There are many ways to add noise, such as random noise, controlled rounding, and multiplicative noise. However, if users are allowed unlimited access to noisy aggregates, data security will be breached eventually. Dinur and Nissim [14] provide a polynomial construction algorithm of data from perturbed subset sums. They show that in order to achieve privacy one has to add perturbation of magnitude $\Omega(n)$, where n is the size of the statistical database. In other words, unless the total number of

queries is sub-linear in the size of the database, a substantial amount of noise is required to avoid breach. Dwork et al. [15] extend the SUM function to general functions f and provide a positive conclusion that privacy can be preserved by calibrating the standard deviation of the noise according to the sensitivity of the function f.

Notes on Recommended Reading

Statistical databases have been an active area of research for over 30 years now, with a significant amount of literature devoted to the topic. Adam and Wortmann [4] provide a comprehensive survey of the classical techniques. Dwork [3] gives a comprehensive survey of the recent work on differential privacy–based approaches.

Recommended Reading

1. Li Y, Wang L, Jajodia S (2002) Preventing interval-based inference by random data perturbation. In: Privacy enhancing technologies. Springer, Berlin, pp 160–170
2. Kenthapadi K, Mishra N, Nissim K (2005) Simulatable auditing. In: PODS. ACM, New York, pp 118–127
3. Dwork C (2008) Differential privacy: a survey of results. In: Agrawal M, Du D, Duan Z, Li A (eds) Theory and applications of models of computation. Lecture notes in computer science, vol 4978. Springer, Berlin/Heidelberg, pp 1–19
4. Adam NR, Wortmann JC (1989) Security-control methods for statistical databases: a comparative study. ACM Comput Surv 214:515–556
5. Ozsoyoglu G, Chin FY (1982) Enhancing the security of statistical databases with a question-answering system and a kernel design. IEEE Trans Softw Eng 8(3):223–234
6. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
7. Denning DE (1983) A security model for the statistical database problem. In: SSDBM'83 proceedings of the second international workshop on statistical database management, Berkeley, CA. Lawrence Berkeley Laboratory, Berkeley, pp 368–390
8. Shoshani A (1997) Olap and statistical databases: similarities and differences. In: PODS '97: proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, New York, NY. ACM, New York, pp 185–196
9. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data 1(1):3
10. Cox LH (1980) Suppression methodology and statistical disclosure control. J Amer Stat Assoc 75:377–385
11. Chin FYL, Ozsoyoglu GÄ (1982) Auditing and inference control in statistical databases. IEEE Trans Software Eng 8(6):574–582
12. Denning DE (1980) Secure statistical databases with random sample queries. ACM Trans Database Syst 5(3):291–315
13. Agarwal R, Srikant R, Thomas D Privacy preserving olap. In: SIG-MOD '05: proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY. ACM, New York, pp 251–262
14. Dinur I, Nissim K (2003) Revealing information while preserving privacy. In: PODS '03: proceedings of the twenty-second ACM SIGMOD-SIGACTSIGART symposium on principles of database systems, New York, NY. ACM, New York, pp 202–210
15. Dwork C, Mcsherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd theory of cryptography conference. Springer, New York, pp 265–284

Stein's Algorithm

►Binary Euclidean Algorithm

Stream and Multicast Authentication

Yvo Desmedt[1], Goce Jakimoski[2]
[1]Department of Computer Science, University College London, London, UK
[2]Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA

Synonyms

Broadcast stream authentication; Multicast stream authentication

Related Concepts

►Authentication Codes; ►CBC-MAC and Variants; ►CMAC; ►Digital Signatures; ►GMAC; ►HMAC; ►MAC Algorithms; ►PMAC

Definition

Streams of data are bit sequences whose length might be large and not known in advance. *Stream authentication* refers to the mechanisms that prevent unauthorized modification of the streams.

Background

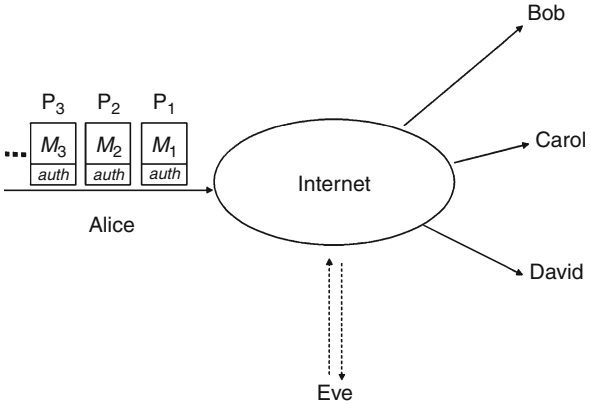
The problem of multicast stream authentication was first studied by Gennaro and Rohatgi [7]. Many schemes have been suggested so far. The Timed Efficient Loss-tolerant Stream Authentication scheme was proposed to be an Internet standard.

Theory

Streams of data are bit sequences that a sender sends to one or more recipients. Unlike the length of a message, the length of a stream might be unknown prior to the transmission, and the streams are usually processed as they "come." Data streams occur naturally when the buffer/memory is shorter than the message or when real-time processing is required.

Stream authentication schemes provide mechanisms that ensure the authenticity of the received data stream. That is, the recipients can verify that the stream they received was really the one that was sent by the sender. A typical multicast stream authentication setting is depicted in Fig. 1. The participants include the sender Alice, the adversary Eve, and one or more recipients. The stream is sent as a sequence of IP packets over the Internet. The sender also computes some authentication data (e.g., authentication tags, signatures, or hash values) and sends this data along with the packets. The recipients use the authentication data to verify that the stream data they received is the original data sent by Alice. The adversary Eve is computationally bounded. However, it is very powerful. It can eavesdrop, modify, drop, and delay packets. It can also corrupt recipients. The goal of the adversary is by cooperating with the corrupt recipients to trick an honest recipient into accepting a forged stream data (i.e., a stream that was not sent by Alice). The stream authentication scheme is secure if the probability that the adversary will succeed is negligible.

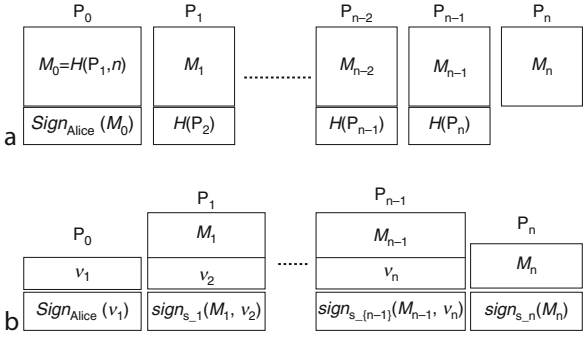
The mechanisms that are used to provide message authenticity (i.e., message authentication and digital signature schemes) cannot be applied in a straightforward manner to provide efficient stream authentication. Suppose the authenticity of the stream packets is protected by using a message authentication scheme as follows. The sender and the recipients are in a possession of a secret key K. Each packet is considered to be a message, and the sender uses the secret key K to compute an authentication tag for each packet. The recipients use the secret key K and the received authentication tag to verify the authenticity of each packet separately. Clearly, an adversary that can corrupt recipients can easily gain access to the secret key and break the stream authentication scheme. A possible countermeasure is to use different keys for different recipients. However, the amount of computation at the sender's side grows linearly with the number of recipients and is prohibitively expensive even for a relatively small number of recipients. One can use a digital signature scheme to compute a digital signature over each packet. Since the recipients will use a public key to verify the authenticity of the packets, the corruption of recipients will have no impact on the security of the schemes. However, this solution is also computationally expensive considering that most stream applications are real-time applications.



Stream and Multicast Authentication. Fig. 1 The multicast stream authentication setting

Two efficient solutions, one off-line and one online, have been proposed by Gennaro and Rohatgi. The off-line scheme assumes that the stream is known in advance. It is depicted in Fig. 2a. The stream is divided into n chunks, and it is processed backwards. The hash of the last chunk n is included in the $(n-1)$-th packet. The hash of the $(n-1)$-th packet is included in the $(n-2)$-th packet, and so on. The hash of the first packet is sent in a digitally signed bootstrap packet. The security of the scheme follows from the unforgeability of the digital signature scheme that is used to sign the bootstrap packet and the collision resistance of the hash function.

The online solution, which is illustrated in Fig. 2b, works for streams that are not known in advance too. The authenticity of the packets is protected by using one-time signatures. To authenticate the contents of the i-th packet, the sender generates a signing (or secret) key s_i and a verifying (public) key v_i in a one-time signature scheme. The public key v_i is sent to the recipients with the previous packet. The first public key v_1 is sent in a bootstrap packet that is signed using a regular digital signature scheme. The recipients use the public key of the sender to check



Stream and Multicast Authentication. Fig. 2 The Gennaro–Rohatgi schemes: (**a**) The off-line scheme, (**b**) The online scheme

the regular digital signature over the bootstrap packet. Once the authenticity of v_1 is verified, the receivers use it to check the authenticity of P_1, and therefore v_2 too. If the one-time signature of P_1 is valid, they use v_2 to check the one-time signature of P_2, and so on.

Various schemes have been proposed subsequently [2–6, 9, 10, 13–17] culminating with the proposal of TESLA [11, 12] for an Internet standard. Although one-time signatures are more efficient than digital signatures, they are not as efficient as message authentication codes. The paradigm depicted in Fig. 3a which has been considered in [1], allows for a multicast stream authentication whose computational cost is about one message authentication code evaluation per packet. To protect the authenticity of the i-th packet, the sender first commits to a key K_i by sending a hash $H(K_i)$ of the key to the recipients. Next, it uses the key K_i to compute an authentication tag over the contents of the packet, and multicasts the packet and the authentication tag to the recipients. Finally, after some delay that guarantees the recipients have received the packet and the tag, the sender reveals the key K_i. To verify the authenticity of the received packet, the receivers check whether the hash of the revealed key is equal to the commitment $H(K_i)$ they have received. If this is the case, they assume that the revealed key is authentic, and they use it along with the authentication tag to verify the authenticity of the packet contents.

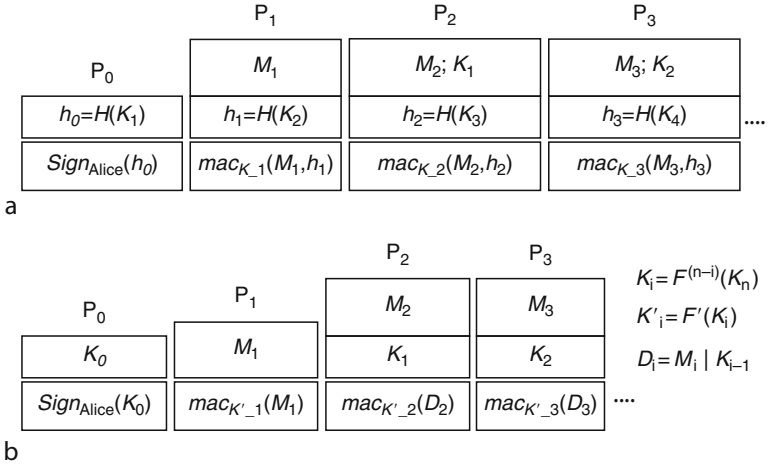
The adversary cannot forge the i-th packet with significant probability, given that:

- The commitment $H(K_i)$ is authentic.
- The commitment $H(K_i)$ does not leak information about the key that will enable the adversary to forge a packet.
- The hash function H is collision resistant (i.e., it is hard to find a key $K_i' \neq K_i$ s.t. $H(K_i') = H(K_i)$).
- The key is revealed only after all the recipients have received the packet.

The first condition is achieved by sending the first commitment $H(K_1)$ in a digitally signed bootstrap packet. Each subsequent commitment $H(K_i)$ is sent in a packet whose authenticity has been verified by the recipients before they use K_i. The subsequent two conditions can be met by carefully choosing the hash function H and the MAC scheme. The last condition is nonstandard and requires time synchronization between the sender and the recipients. Thus, although more efficient, this paradigm provides security in a weaker model than the previously described schemes.

The basic scheme of Fig. 3a is not loss-tolerant. For instance, if the packet P_{i-1} is lost, then the authenticity of the packet P_i and all subsequent packets cannot be verified since the commitment to the key K_i is lost. Furthermore, the packet P_{i+1} that reveals the key K_i has to be sent with a delay (i.e., after all the recipients have received the packet P_i). This condition severely limits the stream transmission rate. The Timed Efficient Loss-tolerant Stream Authentication (TESLA) scheme builds on the basic scheme to address the packet-loss and efficiency issues.

In TESLA, the keys K_i are generated by iterative application of a pseudorandom function to some initial value as illustrated in Fig. 3b. Let $F^v(x) = F^{v-1}(F(x))$ be the transformation defined by v consecutive applications of the pseudorandom function F, and let $F^0(x) = x$. The sender picks uniformly at random some initial key value K_n and pre-computes a key chain, which is a sequence of n key



Stream and Multicast Authentication. Fig. 3 The timed efficient stream loss-tolerant authentication scheme: (**a**) The basic scheme, (**b**) TESLA Scheme II

values $K_0, \ldots, K_{n-1}$, where $K_i = F^{n-i}(K_n), i = 0, \ldots, n$. The key K'_i, which is used to compute an authentication tag for the i-th packet P_i, is derived from the corresponding key value K_i by applying a function F'. The first key in the chain K_0 is multicasted in a digitally signed bootstrap packet. Suppose a given receiver has not received K_{i-2} due to packet loss. Then, it can check the validity of the revealed key K_i by checking whether $K_{i-j} = F^j(K_i)$ given it has received some earlier key K_{i-j} $(j > 2)$. Also, the recipients can recover a lost key K_i by iteratively applying F to some later key value K_j $(j > i)$.

The function F is implemented as $F(K_i) = f_{K_i}(0)$, where f is a target collision resistant pseudorandom function. There are no requirements imposed on the function F' in the original description of TESLA. However, RFC4082 requires $F'(K_i)$ to be computed as $F'(K_i) = f'_{K_i}(1)$, where f' is a pseudorandom function. Suppose that the receivers can decide whether a given packet arrived safely (i.e., before the corresponding key disclosure packet was sent by the sender), then the scheme is secure [8] assuming:

- The digital signature scheme that is used to sign the bootstrap packet is secure.
- The message authentication scheme is secure even when the adversary knows the commitments $K_j, j < i$, to the secret key K_i used by the MAC scheme and
- The function $F(K) = f_K(0)$ is strongly collision resistant.

Applications

Although digital video and audio are the most common streaming applications, streams are quite common in financial applications as well. The volumes of stock quotes, customer-related data, and other market data are growing rapidly and taking the form of continuous streams rather than finite stored data sets. Stream authentication schemes are essential for protecting the integrity of the information in these applications.

Recommended Reading

1. Anderson R, Bergadano F, Crispo B, Lee J, Manifavas C, Needham R (1998) A new family of authentication protocols. ACM Oper Syst Rev 32(4):9–20
2. Bergadano F, Cavagnino D, Crispo B (2000) Chained stream authentication. In: Proceedings of selected areas in cryptography 2000. (Lecture notes in computer science 2012) Springer, Heidelberg, pp 142–155
3. Canneti R, Garay J, Itkis G, Micciancio D, Naor M, Pinkas B (1999) Multicast security: a taxonomy and some efficient constructions, In: Infocom '99. IEEE, Piscataway, NJ
4. Chan A (2003) A graph-theoretical analysis of multicast authentication. In: Proceedings of the 23rd international conference on distributed computing systems, Providence, RI
5. Cheung S (1997) An efficient message authentication scheme for link state routing. In: Proceedings of the 13th annual computer security application conference, IEEE Computer Society Washington, DC
6. Desmedt Y, Jakimoski G (2007) Non-degrading erasure-tolerant information authentication with application to multicast stream authentication over lossy channels. In: The proceedings of the cryptographers' track at the RSA conference 2007. Lecture notes in computer science 4377. Springer, Berlin, pp 324–338
7. Gennaro R, Rohatgi P (1997) How to sign digital streams. In: Proceedings of crypto '97. Lecture notes in computer science, vol 1294. Springer, Heidelberg, pp 180–197
8. Jakimoski G Some notes on the security of the timed efficient stream loss-tolerant authentication scheme. In: The proceedings of Selected Areas of Cryptography 2006. Lecture notes in computer science, vol 4356. pp 345–361
9. Miner S, Staddon J (2001) Graph-based authentication of digital streams. IEEE Symposium on Security and Privacy
10. Park JM, Chong EKP, Siegel HJ (2003) Efficient multicast stream authentication using erasure codes. ACM Trans Inf Syst Secur 6(2):258–285
11. Perrig A, Canneti R, Tygar JD, Song D (2000) Efficient authentication and signing of multicast streams over lossy channels. In: Proceedings of the IEEE Security and Privacy Symposium
12. Perrig A, Song D, Canneti R, Tygar JD, Briscoe B (2005) Timed efficient stream loss-tolerant authentication (TESLA): multicast source authentication transform introduction, internet request for comments, June 2005, RFC 4082
13. Rohatgi P (1999) A compact and fast hybrid signature scheme for multicast packet authentication. In: 6th ACM conference on computer and communications security
14. Syverson PF, Stubblebine SG, Goldschlag DM (1997) Unlinkable serial transactions. In: Financial cryptography '97. Lecture notes in computer science, vol 1318. Springer, Berlin
15. Tartary C, Wang H (2006) Achieving multicast stream authentication using MDS codes. In: The proceedings of CANS'06. Lecture notes in computer science, vol 4301. Springer, Berlin, pp 108–125
16. Wong CK, Lam SS (1998) Digital signatures for flows and multicasts. In: Proceedings of IEEE ICNP '98
17. Zhang K (1998) Efficient protocols for signing routing messages. In: Proceedings of the symposium on network and distributed system security

Stream Cipher

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Symmetric Cryptography

Definition

A *stream cipher* is a symmetric cipher which operates with a time-varying transformation on individual plaintext

digits. By contrast, ►block ciphers operate with a fixed transformation on large blocks of plaintext digits. More precisely, in a stream cipher, a sequence of plaintext digits, $m_0 m_1 \ldots$, is encrypted into a sequence of ciphertext digits $c_0 c_1 \ldots$ as follows: a pseudo-random sequence $s_0 s_1 \ldots$, called the ►running-key or the *keystream*, is produced by a finite state automaton whose initial state is determined by a secret key and a public parameter. The i-th keystream digit only depends on the secret key and on the $(i-1)$ previous ciphertext digits. Then, the i-th ciphertext digit is obtained by combining the i-th plaintext digit with the i-th keystream digit. The most famous stream cipher is the ►Vernam cipher, also called *one-time pad*, that leads to perfect secrecy (the ciphertext gives no information about the plaintext).

Background

Stream ciphers come down to the beginning of the twentieth century with the Vernam cipher [5] and have been widely used for decades. However, the development of different techniques (e.g., linear attacks, correlation attacks, algebraic attacks) at the end of the 1990s led to the cryptanalysis of most stream ciphers, except those derived from block ciphers. This was the motivation of the ►eSTREAM project, which led to the recommendation of several stream ciphers after a 3-year public competition.

Theory

Synchronous and Asynchronous Stream Ciphers

Stream ciphers are classified into two types: ►synchronous stream ciphers and ►self-synchronizing stream ciphers. However, self-synchronizing stream ciphers fell obsolete, mainly because of the existence of a resynchronization mechanism in most modern synchronous stream ciphers.

Families of Stream Ciphers

Stream ciphers can be split into four main families:

- *Provably secure stream ciphers under some computational hardness assumptions*, in the same vein as asymmetric ciphers. For instance, a keystream generator can be constructed by iteratively applying the RSA function from a seed x_0, i.e., $x_{t+1} = x_t^e \bmod pq$, and whose output at time t corresponds to the least significant bit of x_t [1]. Another example is the Blum–Blum–Shub generator [3] whose next-state function is the square function modulo pq.
- *Information-theoretically secure stream ciphers* which aim at providing perfect secrecy like the Vernam cipher, but with some additional assumptions on the capacities of the adversary (e.g., it is supposed that her storage capacity is limited) [2, 4]. These first two families are mainly of theoretical interest, and their costs (in terms of speed, or related infrastructure) preclude their use in practical applications.
- *Stream ciphers derived from block ciphers*: any block cipher used with some appropriate mode of operation is a stream cipher. These modes of operation include the CTR (counter) mode, the OFB (output feedback) mode which lead to synchronous stream ciphers, and the CFB (cipher feedback) mode which provides a self-synchronizing stream cipher. In most applications, block ciphers are actually used as stream ciphers.
- *Dedicated stream ciphers* are specially designed for this usage. They include some ciphers with a large internal state and a nonlinear next-state function which are mainly dedicated to software implementations (e.g., RC4), and some ciphers with a much smaller internal state which are designed for hardware environments (e.g., A5/1).

Families of Attacks Against Stream Ciphers

Most attacks against synchronous stream ciphers are known-plaintext attacks: it is assumed that the attacker knows some keystream bits. These attacks may be turn into ciphertext-only attacks by exploiting some redundancy in the plaintext. Also, the existence of a resynchronization mechanism may be exploited in the context where the attacker knows some keystream bits derived from the same key but with different IVs, including related-IV attacks and chosen-IV attacks. Self-synchronizing stream ciphers are mainly vulnerable to chosen ciphertext attacks.

The attacks against synchronous stream ciphers differ, depending on their objectives. The following classes can be distinguished:

- *Key-recovery attacks*.
- *State-recovery attacks*, which are weaker than key-recovery attacks since they enable the attacker to generate any keystream bit derived from a given initial state, i.e., from given key and IV. But, these attacks do not recover the keystream bits obtained after resynchronization, i.e., after any modification of the IV.
- *Distinguishing attacks*, which determine whether a given sequence is likely to have been generated by the considered stream cipher or whether it is a truly random sequence. Distinguishing attacks are obviously much weaker than key-recovery or initial-state-recovery attacks. They can be used for reducing the uncertainty of unknown plaintexts. Also, it is known that the keystream generator in a synchronous stream cipher passes the *next-bit-prediction test* if and only if it resists all distinguishing attacks [6].

Therefore, it is required that any distinguishing attack against a synchronous stream cipher be not less complex than an exhaustive search for the secret key.

Properties of Stream Ciphers

Stream ciphers have several advantages which make them suitable for many applications. They do not require any padding since they operate on smaller entities than block ciphers. They are also appropriate when buffering is limited since the digits are individually encrypted and decrypted. Moreover, synchronous stream ciphers are not affected by error-propagation. Most notably, they are usually faster and have a lower hardware complexity than block ciphers.

Applications

Stream ciphers are used in many applications. They are especially appropriate in the context of noisy communications when short packets must be encrypted or when the bandwidth is limited. Many applications then use a block cipher with an appropriate mode of operation, like AES-CTR.

However, some applications may require better performance, in particular, some software applications which need a very high throughput, or some hardware applications which need a low-cost implementation (e.g., embedded systems). In all these cases, dedicated stream ciphers offer a well-suited solution.

Recommended Reading

1. Alexi W, Chor B, Goldreich O, Schnorr CP (1988) RSA and Rabin functions: certain parts are as hard as the whole. SIAM J Comput 17:194–209
2. Aumann Y, Rabin MO (1999) Information theoretically secure communication in the limited storage space model. In: Advances in Cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 65–79
3. Blum L, Blum M, Shub M (1986) A simple unpredicable pseudo-random number generator. SIAM J Comput 15:364–383
4. Maurer U (1992) Conditionally-perfect secrecy and a provably-secure randomized cipher. J Cryptol 5(1):53–66
5. Vernam GS (1926) Cipher printing telegraph systems for secret wire and radio telegraphic communications. J Am Inst Electric Engin 55:109–115
6. Yao AC (1982) Theory and applications of trapdoor functions. In: Proceedings of the IEEE 23rd annual symposium on foundations of computer science. IEEE, Chicago, pp 80–91

Strong Authentication

►Challenge-Response Identification

Strong Collision Resistance

►Collision Resistance

Strong Exclusion

►Separation of Duties

Strong Prime

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Integer Factoring; ►Prime Number; ►RSA Digital Signature Scheme; ►RSA Problem; ►Elliptic Curve Method for Factoring

Definition

A strong prime is an integer p such that

- p is a large prime
- $p - 1$ has a large prime factor, denoted r
- $p + 1$ has a large prime factor
- $r - 1$ has a large prime factor

The precise qualification of "large" depends on specific attacks the strong prime is intended to protect against.

Theory

The term "strong prime" was defined in [1]. For a long time, strong primes were believed to be necessary in cryptosystems based on the ►RSA problem in order to guard against two types of attacks: factoring of the RSA modulus by the $p + 1$ and Pollard $p - 1$ factoring methods, and "cycling" attacks. Rivest and Silverman [2] published a paper in 1999 arguing that strong primes are unnecessary in the RSA cryptosystem. There are two points in their argument. First, that the use of strong primes provides no additional protection against factoring attacks, because the ►elliptic curve method for factoring is about as effective as the $p+1$ and the $p-1$ methods (though none is particularly likely to succeed for random, large primes) and is not prevented by the strong prime conditions. Furthermore, the ►number field sieve can factor RSA modulus with near certainty in less time than these methods. (See ►integer factoring for a discussion on factoring methods.) Secondly, they show that cycling attacks are extremely unlikely to be

effective, as long as the primes used are large. Thus, in the current state of knowledge, there is no rationale for requiring strong primes in RSA. A new factoring method might once again make strong primes desirable for RSA, or on the contrary exploit the properties of strong primes in order to factor more efficiently and thus make strong primes appear to be dangerous.

Recommended Reading

1. Gordon J (1985) Strong primes are easy to find. In: Beth et al (eds) Advances in cryptology – Proceedings of EUROCRYPT 84, A workshop on the theory and application of cryptographic techniques Paris, France, 9–11 April, Lecture notes in computer science, vol 209. Springer, Heidelberg, pp 216–223
2. Rivest R, Silverman R (2001) Are 'strong' primes needed for RSA. Cryptology ePrint archive, report 2001/007. http://eprint.iacr.org/

Strong RSA Assumption

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Theory

Let $1 < \tau \in \mathbb{Z}$ be a security parameter. Let $N = pq$ be a product of two random τ-bit primes and let s be an element of the ►group $\mathbb{Z}_N^*$ (►modular arithmetic). The strong-RSA problem is defined as follows:

given (N, s) as input, output a pair $a, b \in \mathbb{Z}$ such that $a^b = s \bmod N$ and $b \pm 1$.

Loosely speaking, the Strong-RSA assumption states that for a sufficiently large τ the strong RSA problem is intractable.

The Strong-RSA assumption was introduced by Baric and Pfitzman [2]. The assumption is used to construct efficient signature schemes that are existentially unforgeable under a chosen message attack *without* the ►random oracle model. One such system is described in [4] and another in [3]. The Strong-RSA assumption is also the basis of several efficient ►group signature schemes [1].

Recommended Reading

1. Ateniese G, Camenisch J, Joye M, Tsudik G (2000) A practical and provably secure coalition-resistant group signature scheme. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 255–270
2. Baric N, Pfitzman B (1997) Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy W (ed) Proceedings of EUROCRYPT '97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 480–494
3. Cramer R, Shoup V (2000) Signature schemes based on the strong RSA assumption. ACM Trans Inform Syst Sec (ACM TISSEC) 3(3):161–185 (Extended abstract in Proceedings of the sixth ACM conference on computer and communications security, 1999)
4. Gennaro R, Halevi S, Rabin T (1999) Secure hash-and-sign signatures without the random oracle. In: Stern J (ed) Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 123–139

Structural Cryptanalysis

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Definition

Structural Cryptanalysis is a branch of *Cryptanalysis* which studies the security of cryptosystems described by generic block diagrams. It analyzes the syntactic interaction between the various blocks, but ignores their semantic definition as particular functions. Typical examples include ►meet-in-the-middle attacks on ►multiple encryptions, the study of various chaining structures used in ►modes of operation, and the properties of Feistel structures or ►substitution–permutation networks with a small number of rounds.

Theory and Application

Structural attacks are often weaker than actual attacks on given cryptosystems, since they cannot exploit particular weaknesses (such as bad differential properties or weak *avalanche effect*) of concrete functions. The positive side of this is that they are applicable to large classes of cryptosystems, including those in which some of the internal functions are unknown or key dependent. Structural attacks often lead to deeper theoretical understanding of fundamental constructions, and thus they are very useful in establishing general design rules for strong cryptosystems.

Subexponential Time

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Computational Complexity; ►Exponential Time; ►*L* Notation; ►*O*-Notation; ►Polynomial Time

Definition

A *subexponential-time* algorithm is one whose running time as a function of the size x of its input grows more slowly than b^x for *every* base $b > 1$.

Theory

Let $T(x)$ be the running time of an algorithm on inputs of size x. The algorithm is said to be subexponential-time if for every constant base $b > 1$, the running time $T(x)$ satisfies

$$T(x) < b^x$$

for all sufficiently large x. Put another way, every ►exponential-time algorithm eventually takes longer than a ►subexponential-time algorithm as x increases. In ►*O*-notation, this would be written $T(x) = 2^{o(x)}$ or $e^{o(x)}$.

A ►polynomial-time algorithm is by definition subexpon ential-time. However, in typical usage in cryptography, subexponential-time refers to algorithms whose running time grows faster than polynomial-time, and more slowly than exponential-time.

(In ►computational complexity, subexponential time sometimes refers to the related notion that for all $o > 0$, $T(x) < 2^{x^{\varepsilon}}$, for all sufficiently large x.)

Applications

Subexponential-time algorithms occur in cryptography in connection with the ►discrete logarithm problem in finite fields and ►integer factoring. The fastest algorithms known for those problems (i.e., the ones whose running times grow most slowly as a function of input size) typically have running times of the form

$$e^{(\gamma+o(1))(\log x)^t(\log\log x)^{1-t}}, \quad \text{for } x \to \infty,$$

for some constants $\gamma > 0$ and $0 < t < 1$, where x is the ►order of the ►finite field in which discrete logarithms are being computed, or the modulus to be factored. The size of the input to these algorithms is proportional to the length in bits of x, so the running time, being subexponential in $\log x$, is subexponential in the input size as well (►L-notation).

Subgroup

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Group; ►Order

Definition

A subgroup is a subset S′ of the elements of a ►group $G = (S, \times)$ such that S' is itself a group with respect to the same group operation.

Theory

If $G = (S, \times)$ is a group, then for any $g \in S$, the set of elements

$$g, g^2, g^3, \ldots$$

(together with the multiplication operation) is also a group. This set is one example of a subgroup of G: a subset of the elements of the group that follows the group axioms (closure, associativity, identity, inverse).

The ►order of any subgroup of a group G divides the order of the group G itself; this is known as *Lagrange's theorem*.

Applications

The ►order and structure of a subgroup in which cryptographic operations are computed can often have a significant impact on the security level. A small-subgroup attack, for instance, exploits the fact that the set of elements generated by a cryptographic parameter is too small and therefore can be searched exhaustively to determine a secret value. In general, subgroups employed in cryptosystems have a large order that is often a prime number or a small multiple of a prime.

Subgroup Cryptosystems

►Torus-Based Cryptography

Subscriber Identity Module

►SIM/UICC

Substitution–Permutation (SP) Network

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers

Definition

Shannon [1] suggested to use several mixing layers interleaving substitutions and permutations to build strong block ciphers. Such design is called a *substitution–permutation sandwich* or a substitution–permutation network (SPN). Although weak on its own, a line of *substitutions* followed by a *permutation* has good "mixing" properties: Substitutions add to local *confusion* and permutation "glues" them together and spreads (*diffuses*) the local confusion to the more distant subblocks (►Substitutions and Permutations).

Theory

If one considers flipping a single bit at the input of such a network, it effects the m output bits of particular S-box which in turn are sent to different S-boxes by a permutation. Thus, inputs/outputs of up to m S-boxes would be effected by the *avalanche* of change. These are again permuted into different S-boxes, covering almost all the S-boxes of the network. On the output of such network, about half of the bits are effected by change and are flipped and about half of the bits are not flipped. This makes an outcome of a single bit change at the input hard to predict, especially if secret key bits are mixed into the block between the layers of encryption. Without a secret key, the SPN performs a complex but fully deterministic function of its inputs.

Applications

Modern ciphers tend to use linear or affine mappings instead of permutations, which allow them to achieve better diffusion in fewer iterations. Such networks are called *substitution–linear* (SLN) or *substitution–affine networks* (SAN). The block encryption standard Rijndael/AES is an SLN cipher. Designers of AES have formalized Shannon's and DES team's confusion-diffusion ideas into a *wide trail design strategy* [2], which allows the designer to prove lower bounds on probabilities of differential or linear trails.

Recommended Reading

1. Shannon CE (1949) Communication theory of secrecy system. Bell Syst Tech J 28:656–715
2. Daemen J, Rijmen V The design of rijndael. Springer, Berlin, pp 123–147

Substitutions and Permutations

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Alphabet; ►Encryption; ►Symmetric Cryptosystem

Definition

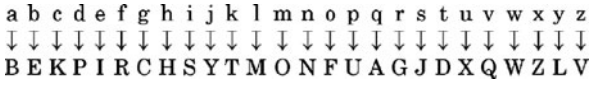
A substitution cipher is usually described by a sequence or list of single substitutions, each of which is commonly denoted by an arrow, like $p \mapsto \pi$.

Example: The Russian-English ISO transliteration (using diacritical marks) is a substitution.

A substitution may have homophones (►Encryption).

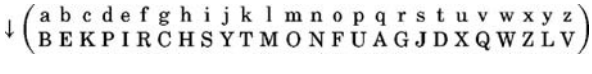
A *permutation* is a one-to-one mapping from an ►alphabet to itself.

A substitution may be described by two lines: the first one being the standard alphabet and the second one being a mixed alphabet (►Alphabet). An example is given below:



Note that small letters are used for the plaintext and capital letters for the ciphertext.

In mathematics, there is a commonly used, simplified notation with two lines bracketed together:



This is convenient for ►encryption. For decryption, it is worth while to rearrange the list:

$$\uparrow \begin{pmatrix} \text{q a g t b o r h e s c y l n m d v f i k p z w u j x} \\ \text{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z} \end{pmatrix}$$

or

$$\downarrow \begin{pmatrix} \text{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z} \\ \text{q a g t b o r h e s c y l n m d v f i k p z w u j x} \end{pmatrix}$$

There is also the cycle notation which is shorter

(a b e i s j y l m o f r g c k t d p u x z v q) (h) (n) (w)

but this notation is inconvenient both for encryption and decryption. The cycle is generated by iterating the substitution on an arbitrarily chosen starting letter; whenever a cycle is closed, a new starting letter is chosen until all letters are exhausted.

Self-reciprocal permutations are permutations that, when applied twice, restore the original. Put equivalently, they are their own inverse. Their cycle notation shows decomposition in two-cycles and one-cycle, for example:

(an)(bx)(ds)(ei)(fv)(gh)(ku)(lc)(mq)(ow)
(py)(j)(t)(z)

If a self-reciprocal permutation has no one-cycle (so n is even), there is also the following notation

$$\updownarrow \begin{pmatrix} \text{a b c d e f g h i j k l m} \\ \text{n o p q r s t u v w x y z} \end{pmatrix}$$

The Enigma machine of the German *Wehrmacht* used a (properly) self-reciprocal permutation. This was thought to be particularly practical since the same machine could be used for encryption and decryption, disregarding the fact that this opened ways for a cryptanalytic attack ("Non-coincidence Exhaustion" in ►Cryptanalysis).

A *substitution cipher* in general replaces certain groups of characters by certain other groups of characters. This may be described by a list, e.g., for $\mathbb{Z}_2^3 \to \mathbb{Z}_2^3$:

$$(000) \longmapsto (001),\ (001) \longmapsto (010),\ (010) \longmapsto (011),\ (011) \longmapsto (100),$$
$$(100) \longmapsto (101),\ (101) \longmapsto (110),\ (110) \longmapsto (111),\ (111) \longmapsto (000).$$

Some more terms that one may see in this context are given here. A *monographic* substitution is a substitution *of* single characters, while a *unipartite* substitution is a substitution *by* single characters.

A *simple* substitution is a substitution of single characters by single characters, so it is a monographic, unipartite substitution.

A *digraphic* substitution is a substitution *of bigrams* (ordered pairs of characters). A *bipartite* substitution is a substitution *by* bigrams. Finally, a *bigram* substitution is a substitution of bigrams by bigrams, so a digraphic, bipartite substitution.

In general, an *n-graphic* substitution is a substitution *of* n-tuples of characters (n-grams) and an *n-partite* substitution is a substitution *of* n-tuples of characters. Similarly, a *polygraphic* substitution is an n-graphic substitution, $n \geq 2$, and a *multipartite* substitution is an n-partite substitution, $n \geq 2$. A *linear substitution* is a block encryption $\mathbb{Z}_N^n \to \mathbb{Z}_N^m$ that is the composition of a translation t and an homogenous part φ which is additive with respect to addition modulo N (for all $x, y \in \mathbb{Z}_N^n : \varphi(x + y) = \varphi(x) + \varphi(y)$).

A *null* is meaningless ciphertext character, the encryption image of the empty plaintext word. It is used, e.g., for swamping the plaintext statistics or masking the occurrence of idle times.

A *straddling encryption* or *straddling cipher* is a substitution with encryption steps $V^{(l)} \to W^{(m)}$, where $Z^{(k)}$ denotes the set of all sequences of at most k characters from Z, in formula $\{\varepsilon\} \cup Z \cup Z^2 \cup Z^3 \ldots \cup Z^k$, where Z^n is the set of all words of length n over the alphabet Z, and ε denotes the empty word.

Example: $Z_{20}^{(3)} \to \mathbb{Z}^{(2)}$ with the homophonic substitution

$$\downarrow \begin{pmatrix} \text{che} & \text{con} & \text{non} & \text{et} & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\ 44 & 64 & 00 & 08 & 1 & 86 & 02 & 20 & 62 & 22 & 06 & 60 & 3 \\ & & & & & & & & 82 & & & & \end{pmatrix}$$

$$\downarrow \begin{pmatrix} \text{l} & \text{m} & \text{n} & \text{o} & \text{p} & \text{q} & \text{r} & \text{s} & \text{t} & \text{v} & \text{z} & \epsilon \\ 24 & 26 & 84 & 9 & 66 & 68 & 28 & 42 & 80 & 04 & 88 & 5 \\ & & & & & & & & 40 & & & 7 \end{pmatrix}$$

Both 5 and 7 are in this example (Matteo Argenti, 1590) nulls. Other elements of $Z_{20}^{(3)}$ have no image, except by composition of their individual letters.

Let a *block* be a text of predetermined length. Then a ►block cipher or *block encryption* is a substitution with encryption steps $V^n \to W^m$, i.e., without straddling. The block length is usually rather high (for instance, the ►Data Encryption Standard has a block length of $m = n = 64$, and the *Advanced Encryption Standard* (►Rijndael/AES) has a block length of $m = n = 128$, 192, or 256 bits). The same block encryption step with its key is repeated on and on, thus, each bit of ciphertext in a given block normally depends on the complete corresponding plaintext block,

with as consequence the possibility of error propagation over the full block.

A ►stream cipher (also called *stream encryption*) is a substitution $(V^n)^* \rightarrow (W^m)^*$ between infinite series of blocks, controlled by a ►key generating algorithm. The generated key may have a finite period. Autokey or other cipher feedback is excluded.

A *transposition cipher* or *tranposition* does not substitute the characters of a message, but permutes their position; it may be considered as a special case of a polygraphic substitution $V^n \rightarrow V^n$ of the kind

$$(x_1, x_2, \ldots, x_n) \longmapsto (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}),$$

where π is a permutation of the *subscripts* $\{1, 2, \ldots, n\}$. It can be performed by multiplication of $(x_1, x_2, \ldots, x_n)$ with a *permutation matrix*, i.e., an $n \times n\{0,1\}$-matrix such that in every row and in every column, one occurs just once. This extreme property makes cryptanalysis of transposition ciphers very different from cryptanalysis of normal substitution ciphers and explains why alternating composition of substitutions and transpositions ("Pastry Dough Mixing" below) is so effective.

A *grille* is a tool, usually in the form of punch cards, which can be rotated to perform a transposition of the letters.

Pastry dough mixing stands for a composition of alternating substitutions and transpositions. It was already recommended by Shannon in 1949 and used, e.g., in the DES cryptosystem. The expression "pastry dough mixing" was introduced by Eberhard Hopf in the mathematical theory of compact spaces.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Summation Generator

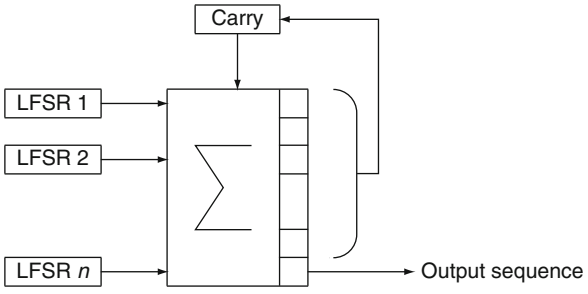
Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI, CNRS/Lab-STICC/CID and Telecom Bretagne, Brest Cedex 3, France

Related Concepts

►Combination; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

The summation generator is based on a ►combination of n ►Linear Feedback Shift Registers (LFSRs), as depicted in Fig. 1, and was first proposed in [1, 2]. The combining function is an addition over the set of integers. From a binary point of view, it is a nonlinear function, with maximum ►correlation immunity. The output bit is the least significant bit of the integer sum.



Summation Generator. Fig. 1 The summation generator

Theory

Powerful attacks exist in the case $n = 2$ [3, 4]. Hence, it is better to use several LFSRs, with moderate lengths, than just a few large ones. But it has also been shown that this scheme is vulnerable if all the LFSRs are short [5]. A ►Fast Correlation Attack has been presented in [6].

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers, Springer, Berlin
2. Rueppel RA (1986) Correlation immunity and the summation generator. Advances in Cryptology – Crypto'85. Lecture notes in computer science, vol 218. Springer, pp 260–272
3. Meier W, Staffelbach O (1992) Correlation properties of combiners with memory in stream ciphers. J Cryptol 5:67–86
4. Dawson E (1993) Cryptanalysis of summation generator. Advances in Cryptology – Auscrypt'92. Lecture notes in computer science, vol 718. Springer, pp 209–215
5. Klapper A, Goresky M (1995) Cryptanalysis based on 2-adic rational approximation. Advances in Cryptology – Crypto'95. Lecture notes in computer science. Springer, pp 262–273
6. Golic J, Salmasizadeh M, Dawson E (2000) Fast correlation attacks on the summation generator. J Cryptol 13:245–262

SWP

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Synonyms

Single wire protocol

Related Concepts
►NFC; ►SIM/UICC

Definition
The Single Wire Protocol is a single wire communication between a host controller and a contactless front end according to the [ETSI TS 102 613] standard.

Background
In, for example, mobile phones, the contactless front end and the host processing are performed by two different chips. SWP is a protocol standardizing the communication between those two chips.

SWP has been defined for the ETSI SCP (= Smart Card Platform) support.

Recommended Reading

1. Finkenzeller K (2010) RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, 3rd edn. Wiley, Chichester, West Sussex
2. Single wire protocol (SWP), 2007, ETSI. *ETSI TS 102 613 standardizes the Single Wire Protocol to be used between an Contactless Front End and a UICC.*
3. UICC-contactless front-end interface, Host controller interface (HCI), 2008, ETSI. *ETSI TS 102 622 standardizes the architecture of the Host Controller Interface between the UICC and the Contactless Front End.*

Symmetric Cryptosystem

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Synonyms
Classical cryptosystem; Conventional cryptosystem; Private key cryptosystem; Secret key cryptosystem

Related Concepts
►Asymmetric Cryptosystem; ►Block Ciphers; ►MAC Algorithms; ►Stream Cipher

Definition
In a *symmetric cryptosystem*, the same ►key is employed for each of the operations in the cryptosystem (e.g., encryption and decryption), and thus that same key, typically a secret, must be shared by the parties performing the various operations.

Applications
Symmetric cryptosystems are widely employed; see the related concepts for specific examples.

SYN Cookie Defense

Wesley M. Eddy
Verizon/NASA Glenn Research Center, Cleveland, Ohio, USA

Synonyms
TCP SYN cookies

Related Concepts
►SYN Flood Attack

Definition
The Transmission Control Protocol (TCP) SYN Cookie Defense is a technique for mitigating the effects of TCP SYN flooding attacks. The strengths of the SYN cookie defense are that it eliminates the listening server's need to maintain state for half-open connections or timers for reaping that state, and it only requires direct support within the listening server's TCP implementation and zero modification to the initiating client's TCP implementation.

Background
After the initial publicity around the TCP SYN flooding attack, several mitigation techniques were developed, implemented, and discussed. One of the most well known of these mitigations is the SYN Cookie technique. This is also historically one of the more controversial techniques, as there had been some amount of misinformation about it circulating, and emotional arguments resulted.

The basic idea behind SYN cookies is to postpone the allocation of state on a listening server until after the completion of TCP's three-way handshake. Normally, some amount of state would be allocated upon receiving a SYN segment (step 1 of the three-way handshake), so that later when an ACK segment is received (step 3 of the three-way handshake), it can be validated. Because this can make implementations vulnerable to a SYN flooding attack, in which case ACKs are never received, the SYN cookies technique takes the state that normally would be allocated and encodes it into fields of the SYN-ACK that can be validated and reconstructed from the later acknowledgement (ACK) segment generated by legitimate hosts trying to connect. In this way, attackers elicit a SYN-ACK, but cause no other

resource consumption on the victim, and legitimate connections proceed as normal, even when the machine is under heavy attack.

Theory

There are multiple approaches to defending against TCP SYN flooding attacks; some are based on adding protection within the network infrastructure, and others are based on modifying end-host TCP implementations. Several deployed mitigation techniques have been described by Eddy [1]. Among the end-host modifications, there are simple but problematic tactics involving simple tuning parameters like the connection backlog used with the *accept* system call, or reducing the timer for the SYN-RECEIVED state. The more-effective techniques involve modifying the implementation behavior on receiving a SYN in order to defer resource allocation until a later ACK is received. Two approaches in this class include SYN caches and SYN cookies.

The SYN cache approach, as described by Lemon [3], stores partial connection state information for SYN-RECEIVED connections in a hash table after receiving a SYN, and then matches ACKs up against the hash table entries in order to flesh them out into fully ESTABLISHED connection state structures after a legitimate TCP handshake completes. The SYN cache approach still involves allocating some memory for the hash table, though it can be fixed in size, and also management of the hash table does incur some amount of overhead.

SYN cookies, as proposed by Dan Bernstein, are a variation of the cookie approach used in other protocols to completely delay allocation of state until after a handshaking process has completed to vet both sides of the connection. All the essential states that would normally have been stored after receiving the SYN are instead reconstructed from the ACK that completes the handshake. This can be done through clever encoding of the connection state data into certain fields of the SYN-ACK that are echoed in the later ACK as part of the TCP protocol (e.g., the sequence number field in the SYN-ACK is recoverable from the acknowledgment field in the ACK, and the time stamp within the SYN-ACK is returned in the time-stamp-echo field of the ACK).

There are multiple implementations of the SYN cookie concept, the details of which vary between implementations. Some are limited in not being able to support advanced TCP options (like Selective Acknowledgement and Window Scaling), while others are less limited. These limitations stem from the small number of bits available to encode information into within the TCP header fields that are echoed from the SYN-ACK within the ACK. Implementations that only use the sequence number to hold the cookie are less capable than those that are able to also use the time-stamp field, but the time-stamp field is also only possible to use when the corresponding party also supports normal use of TCP time stamps, which are not part of the base TCP protocol, but are a widely supported optional extension.

The basic contents of a SYN cookie that multiple implementations have included consist of three parts: (1) an index into a pool of local secret bits, (2) an encoding of an approximate maximum segment size bound (needed for TCP to operate without fragmentation), and (3) a number of bits computed via truncation of a hash function output. The hash function is computed over the local secret bits identified by the index, the connection-identifying information within the SYN-ACK packet (IP addresses, TCP port numbers, etc.), and the sequence number that the SYN contained. When the cookie is echoed back, these hash function inputs can all be obtained from the incoming ACK packet and evaluation of the hash function is used to verify the legitimacy of the cookie within the ACK.

In general, it is expected that the probability of guessing a valid SYN cookie to include in an ACK is near 2^{24} (when the hash output is truncated to 24-bits, as in common implementations), due to the use of local secret bits and the whitening of the IP adddresses, TCP port numbers, etc. through the hash function. In practice, several implementations to date have used MD5 as a hash function, and some have used SHA-1 to perform the hash.

A major advantage of SYN cookies is that they only need to be supported on the host running the listening server application. No modifications to client systems are necessary, as the fields used to hold cookie data are by design ones that can be recovered from the ACK.

Applications

Today, various forms of SYN cookies have been implemented in several operating systems, including FreeBSD and Linux. Some TCP connection-proxying gateway devices are also available that split TCP connections and provide defense for a network of hosts behind them by implementing SYN cookies at the proxy. The presence of SYN cookies in some common TCP implementations prompted the description and analysis edited by Eddy [2].

In addition to TCP implementations that use SYN cookies, other more recently designed protocols also employ cookie mechanisms. The Stream Control Transmission Protocol (SCTP), for example, uses a four-way handshake, and only allocates session state at the listener

after the initiator has echoed back a cookie. Similarly, the Internet Key Exchange version 2 (IKEv2) protocol does not create a security association until legitimacy of peers have been verified through a multi-packet exchange, creating a similar resilience to flooding of connection attempts.

Open Problems

Today, several techniques for mitigating SYN flooding attacks seem to be well accepted and somewhat well deployed, including SYN cookies present in a number of end-host TCP implementations and specialized network perimeter security devices. One open aspect of the problem is that of protecting mobile devices from TCP flooding attacks, as described by Swami and Tschofenig [4], who observed that the attack and defenses have different implications in wireless or mobile networks, and proposed a unique mitigation for these environments.

Recommended Reading

1. Eddy W (2006) Defenses against TCP SYN flooding attacks. Cisco Internet Protocol J 9(4)
2. Eddy W (2007) TCP SYN flooding attacks and common mitigations. RFC 4987
3. Lemon J (2002) Resisting SYN flood DoS attacks with a SYN cache. BSDCON 2002
4. Swami Y, Tschofenig H (2006) Protecting mobile devices from TCP flooding attacks. Proceedings of the first ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch), December 2006

SYN Flood Attack

Wesley M. Eddy
Verizon/NASA Glenn Research Center, Cleveland, Ohio, USA

Synonyms

TCP SYN flooding

Related Concepts

►SYN Cookie Defense

Definition

The SYN flooding attack is a denial-of-service method that exploits the design of the Internet's Transmission Control Protocol (TCP) three-way handshake for establishing connections by exhausting a server's allocated state for a listening server application's pending connections, preventing legitimate connections from being established with the server application.

Background

SYN flooding attacks were first widely publicized by a *Phrack* magazine article in 1996 [1], and also began being witnessed on the Internet.

The attack exploits the common policies of operating system TCP implementations in enforcing a maximum number of TCP connections in the SYN-RECEIVED state for a particular listening server application. By rapidly sending TCP SYN segments to a server, the attacker causes the number of connections in the SYN-RECEIVED state to reach the maximum, and incoming connection requests from legitimate clients are subsequently denied until the attacker's connections expire. The attacker can periodically send more SYN segments in order to retain his control on the server's connection resources.

The attack has been known since the mid-1990s and several defenses have been developed and deployed. One well-known defense is the use of SYN Cookies (described in another article of this publication), though several other defenses are also popularly used, as described in RFC 4987 [3].

Theory

The SYN flooding attack exploits the common TCP implementation behavior of allocating a connection state structure upon receipt of a TCP SYN segment, while also limiting the number of connections in a particular subset of states for a given application. The size of the connection state structure, often called a transmission control block (TCB), varies between implementations, but in some popular operating systems can range from 280 to over 1,300 bytes, depending on supported TCP features and other implementation decisions. A "backlog" is commonly implemented to bound the amount of host memory used by connection state structures for certain connection states, generally resulting in common applications and host configurations supporting only a few dozen in-progress connection attempts with a particular application.

The SYN flooding attack differs from common packet flooding attacks, in that instead of overloading the network's resources, it merely exhausts the backlog of TCP connection state data structures allocated for a victim application. By an attacker sending SYN segments to the application, but not responding to the SYN-ACK segments generated in response, the victim TCP implementation is forced to slowly time-out instances of connection state

generated by the attacker. Meanwhile, the application is unable to accept connections from legitimate users.

The attack can be tuned for effectiveness by sending a burst of only the number of segments necessary to exhaust the backlog, and no more. This can be done using very low network capacity, since SYN segments are small, and is less easily detectable than a bulk packet flooding attack. The attack can also be tuned to repeat the burst of segments at the same period as the victim TCP implementation times out and reclaims TCBs. The effectiveness of the attack is easy for the attacker to measure by viewing the responses to its SYNs, such as the SYN-ACK bit combination, reset bit (RST), or other indications of failure.

Variations of the attack as well as analysis of several defense techniques are described by Eddy [2].

Applications

The SYN flooding attack has been observed in the wild. A well-publicized early case was in September of 1996 when the Panix ISP's mail servers sufferred outages due to SYN flooding attacks. The sequence of attack packets is not difficult to generate, and other instances of the attack have also been observed.

Experimental Results

Proof-of-concept code implementing the attack followed the *Phrack* article [1], and was widely distributed. The attack has been used, and positively identified numerous times on the Internet since then. It is very easy to reproduce and its effectiveness is well established, however, later experimental results with various defense techniques also show that it can be mitigated effectively, as overviewed by Eddy [2, 3].

Open Problems

Though many defenses to this attack have been implemented and are used in practice, there is no standardized defense mechanism incorporated in either the base TCP protocol specification, or the requirements for Internet hosts maintained by the Internet Engineering Task Force (IETF). The implemented defenses vary on an individual system-by-system basis. A standardized defense or set of recommendations is one possible direction for future work.

Recommended Reading

1. daemon9, route, and infinity, Project Neptune, Phrack Magazine, 7(48) file 13 of 18, July 1996
2. Eddy W (2006) Defenses against TCP SYN flooding attacks. Cisco Internet Protocol J 9(4)
3. Eddy W (2007) TCP SYN flooding attacks and common mitigations. RFC 4987

Synchronous Stream Cipher

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Stream Cipher

Definition

A synchronous stream cipher is a stream cipher, in which the keystream is generated independently of the plaintext and of the ciphertext. The keystream is usually produced by a pseudorandom generator, parameterized by a key, which is the secret key of the whole scheme.

Background

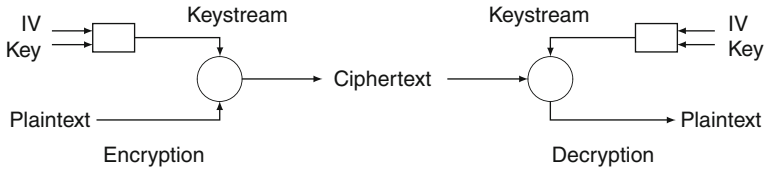
In modern stream ciphers, the initial state of the keystream generator is obtained not only from the key but also from a public initialization vector IV. This IV vector is changed for each new frame encryption, and can be transmitted with no specific protection. Hence, a synchronous stream cipher can be depicted as in Fig. 1.

In a synchronous stream cipher, it is impossible to dynamically check the synchronization between the keystream and the message. The keystreams generated by the sender (encryption) and by the receiver (decryption) must be perfectly synchronized. If synchronization is lost, then decryption fails immediately. If one wants to be able to resynchronize both signals, one needs some additional techniques. A classical solution is to use a new keystream, produced by the same key but a new IV. Hence, the initialization process which changes the IV is called the *resynchronization mechanism*.

Another important property of synchronous stream ciphers is that errors on the ciphertext do not propagate: if the ciphertext is altered by some errors, then this will only affect the decryption of the wrong symbols, but this will have no effect on the others.

Theory

These two properties (perfect synchronization needed, no propagation of errors) lead to some active attacks: the first one could be to modify the ciphertext in order to desynchronize the message and the keystream during decryption (this can easily be achieved by deleting or inserting some bits, for example); the second one consists in modifying



Synchronous Stream Cipher. Fig. 1 Principle of a synchronous stream cipher

the values of some bits in order to modify the plaintext obtained after decryption (this can be powerful if the attacker knows sufficient information about the message in order to choose the meaning of the modified plaintext). This implies that it is important to use, at the same time as encryption, some integrity/authentication techniques in order to avoid such attacks.

Of course there are also attacks targetting the keystream generation process, and attacks targeting the resynchronization process. These kind of attacks are specific to each synchronous stream cipher.

A good starting reference on the topic is [1].

Applications

Most of the stream ciphers used nowadays (see, for example, ►E0 and ►A5/1) are *binary additive stream ciphers*; they are synchronous stream ciphers, in which all the data (plaintext, keystream, ciphertext) are binary, and that simply add (through the XOR function) the message (plaintext/ciphertext) to the keystream.

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers. Springer, Berlin