Authentication Tokens: Balancing the Security Risks with Business Requirements

Joe Grand
kingpin@atstake.com

@stake, Inc.
196 Broadway
Cambridge, MA 02139
www.atstake.com

September 19, 2001

Stolen passwords represent a significant threat to today's enterprise. It has become apparent that a simple username and static non-changing password combination to login to a system is not adequate to protect most business information. As the corporate network is increasingly used to store disparate levels of company confidential information, there is a need for user access control. The theft of a user's password or other static credentials is succeeded by network monitoring, social engineering, or other means.

The classic problems of user authentication are: How can you prove you are who you say you are? Is the user accessing data or logging into the system *really* the person who has been granted the right to do so? Authentication tokens attempt to help solve these problems by reducing the risk of static password sniffing and other spoofing or cloning threats. The use of authentication tokens has replaced the simple username/password model in a wide variety of network infrastructure configurations, ranging from fine-grain control to ubiquitous network access.

Tokens are hardware or software devices that generate dynamic one-time passwords through the use of a mathematical function. Passwords generated

[*]@stake RR2001-04

[†]Product and company names mentioned in this article may be trademarks of their respective owners.

by tokens are different each time the user requests one, so an intercepted password is useless as it will never be used again. Along with the password generated by the token device, the user can still provide a username and possibly an additional static password or personal identification number (PIN). Token devices come in many shapes and sizes, ranging from key chain- to calculator-sized hardware devices to software programs running on personal digital assistants (PDAs) or desktop computers.

The security risk scenarios vary depending on token technology (hardware v. software), business needs, and implementation/deployment strategies. Consider a scenario in which software-based authentication tokens are deployed and become compromised by an attacker who is now able to generate the proper authentication to login as the legitimate user:

Each employee at Company X has a PDA (such as a Palm or PocketPC device) or access to a desktop or laptop computer. This is convenient for the company since it is not necessary to distribute a dedicated hardware device to each user. To deploy a software-based solution, the system administrator generates a specific configuration file for each user. This typically contains the username, PIN, a secret data component used in the mathematical algorithm to create the one-time-passwords, and a unique identification value to enable the server to match the user to the system. The configuration file is given to the user by the administrator and loaded onto the PDA or computer. The data in the configuration file should only be known by the token software and the server. If the configuration file is obtained by an attacker, the authentication token can be cloned, therefore compromising the identity of the legitimate user. Continuing with this example, one employee leaves his unprotected PDA on his desk while he takes a coffee break. A malicious insider, noticing that the employee has left, copies the configuration file to his own PDA (which could be achieved using the infrared port of the device), taking only a matter of seconds. On software-based systems, it is trivial to retrieve data through either standard functionality or common debugging techniques. The attacker now has a copy of the legitimate user's configuration file and can retreat to another location to perform additional analysis to clone the token. This entire attack goes undetected by the legitimate user, who notices nothing different with his PDA. However, it is now possible for the attacker to generate the proper credentials identical to the legitimate user.

1 Business Analysis

Security product vendors are heading towards a slippery slope in which marketing and industry fads often play a larger role than security and business purpose in the design process. If the risks of using software-based token devices cannot be managed or mitigated by either the vendor or user, the product should not be used in the environment presenting those risks. Hardware-based systems are generally considered by security professionals to be capable of greater security than their software counterparts. However, a system that is capable of being more secure does not always mean it is.

1.1 Managing Risk: What Is Being Protected?

Threat modeling and risk assessment are the first steps in deciding on a particular security solution [12]. To decide upon a hardware- or software-based technology requires a thorough understanding of the attack profile and needs of the business. Some questions to ask when considering an authentication token technology are as follows:

- Why is this technology being deployed for the business?
- What is the current access control methodology employed, if any?
- What is the type of access or data being protected?
- How does this type of authentication technology impact the business?
- Are there to be remote users outside of the trusted network?
- What are the perceived threats? Internal or external?
- Are there plans for future expandability?
- Are there any other methods of access control needed (through which hardware devices can be consolidated)?

Tony Walker, Vice President of Development at CRYPTOCard Corporation, comments that "there is a strong market demand for this [software] type of device. We do point out to our customers that these tokens are inherently weaker than the hardware tokens, but many customers choose to use them anyway because of the convenience they offer. As you are well aware, all security is a matter of cost – the cost of breaking it versus the value of the material obtained. It is up to the individual customer to evaluate the trade-offs and make this choice.” [7]

It is indeed necessary to understand what data or system infrastructure needs to be protected before a choice of authentication token technology is decided upon. For example, are hardware tokens being used to protect a simple stand-alone computer where the cost of data loss will be minimal? Are software tokens being used to protect critical portions of your network infrastructure? The trade-offs and middle ground lie between the sensitivity of the data being protected and the likelihood of attack on the actual token device. Deciding on an authentication technology based solely on employee preference or ease-of-use without considering the risks is not addressing the business concern.

“When deploying any security system, there are often trade-offs between security and convenience. Organizations must balance the value of the information being protected with ease-of-use and cost issues to deploy a system that is appropriate for their needs. The reality is that a system that is ‘secure enough’ and used by a large number of people, may be more valuable than a highly secure system that is only implemented by a few. Because organizations have different levels of risk, multiple solutions are needed with varying levels of security strength,” said Willy Leichter, Product Marketing Manager at Secure Computing. [8]

For infrastructures requiring high levels of security, token vendors are recommending the more expensive hardware tokens. For environments that do not justify this expense, software tokens are often chosen.

2 Cost and Deployment Analysis

Deployment cost of authentication tokens can be divided into three stages comprising the Total Cost of Ownership (TCO):

- **Stage 1:** Immediate cost (initial deployment)
- **Stage 2:** Support/Maintenance cost (on-going)
- **Stage 3:** Remediation cost (e.g., revoking and reissuing of token devices after an attack or loss of device)

Figure 1 shows cost impact over time, with the three stages identified.[1]

The price structure will vary depending on vendor and product type, but usually hardware tokens are significantly more costly than software tokens, due to the fact that they are physical products that require their own manufacturing and testing processes. The price of software tokens will often be significantly less because no special hardware is required. This comes at a potential cost of lower overall tamper resistance.

In a large-scale corporation of 10,000 employees, initial deployment costs (Stage 1) including access control software and server-side infrastructure could run upwards of $1 million. For a typical hardware token deployment this example will assume a cost of $80 each, which is an average calculated from multiple vendors' token prices. For a typical software token deployment, the cost-per-token is smaller ($60 on average), since the token application is simply a piece of software running on a device the user already owns (such as a laptop or PDA). This translates to a $200,000 reduction in token device cost for this scenario based solely on technology selection.

On-going support costs (Stage 2) remain fairly consistent between token technologies. The server-side software remains the same regardless of token technology used. Failures with a hardware or software token can simply be fixed by replacing the device or application. The maintenance cost of replacing a broken hardware token is negligible due to product warranties offered by the token vendors (upwards of 5 years).

One major cost area is related to remediation and redeployment (such as a company's change in token technology after there has been a security breach or after a device has been lost). Replacing a lost hardware token may entail paying full price for a new token compared to a software environment where the application can simply be reinstalled onto the system. The risk of losing a software-based token (running on a desktop, laptop, or PDA) is arguably less than losing the small, keychain-sized hardware devices.

A redeployment example (Stage 3) may consist of switching to a hardware-based technology after a successful attack on the software-based de-

[1]Some token vendors are beginning to offer "Token Deployment Services" to provide deployment, configuration, and maintenance support to the end-user. Because it is not known how these services affect TCO and time issues, they are not taken into account in our analysis. We assume the end-user organization has total responsibility of the system.

vices. One could argue that if a single software token has been compromised, others are likely to follow because of the simple, repeatable nature of software attacks. This puts the entire authentication system at risk. If software tokens are deployed first and a switch is then made to hardware tokens, the average cost of the authentication token system increases. By using hardware tokens from the start, the software-related security problems become moot; hence, the risk of successful attack becomes much smaller (Section 3 examines this claim in detail). There is the additional cost of deploying hardware tokens from the beginning with the trade-off that hardware token technologies are not as frequently attacked as their software counterparts. This is a classic example of risk management.

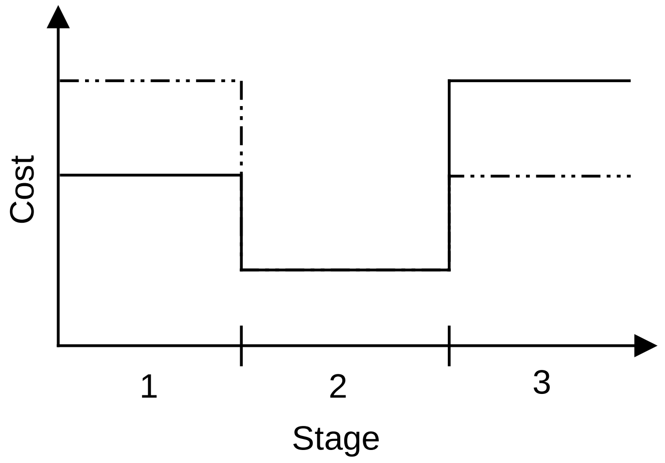

Figure 1: Deployment cost impact for each stage – hardware (dashed) v. software (solid).

2.1 Time Requirements for Deployment

Software-based tokens have an advantage over hardware-based tokens due to the time necessary to deploy. Deployment time for software-based tokens can be considerably reduced, especially if the software platform is already being used by most or all of the employees (e.g., laptops or PDAs). Many times, deploying software-based tokens is as simple as the administrator or user installing the client application software on the device and the administrator creating a configuration key for each specific user. The configuration key, often containing the user-specific identification and credentials, can be distributed electronically (e.g., secure/encrypted e-mail or a central software distribution site internal to the corporation) in minimal time. There is small, if any, disruption to the organization if roll-out is handled efficiently.

Deploying a hardware device requires a physical interaction with every employee. Due to scheduling conflicts, multiple office sites, and employee travel, it may constitute a substantial time investment to meet with each employee and deploy the hardware-based tokens. The larger the corporate infrastructure and user base, the longer hardware deployment will take. For a software-based system, the deployment time will remain much more constant, since the time to meet with each employee is not required and the configuration keys can be distributed electronically. Software-based configuration keys can also be deployed in a face-to-face manner as with hardware tokens, which would require the same significant amount of time. However, a face-to-face distribution avoids using the current network infrastructure which could be insecure (and may be the reason an authentication system is being implemented).

Figure 2 submits that the amount of time required for face-to-face deployment of a hardware- or software-based system will increase linearly as the number of users grows, largely due to the need to physically meet each employee and distribute the tokens or configuration keys.

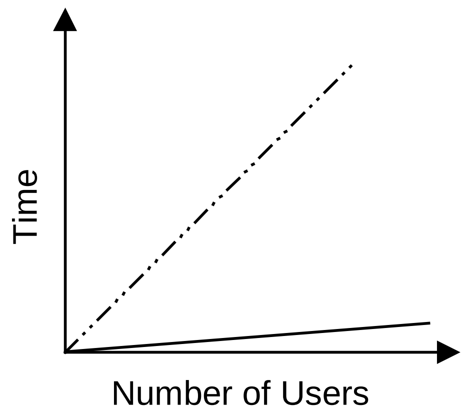

Figure 2: Deployment time – hardware/software face-to-face (dashed) v. software electronic distribution (solid) after initial system setup and configuration.

Management time (initial server setup, configuration, and on-going support) will be similar for either token technology and depends more on the ease-of-use of the vendor's administration interface. Management time may also increase due to the size of the deployment.

3 Security Analysis

Software applications lack the protective mechanisms often found in dedicated hardware devices (e.g., tamper proofing and physical encapsulation of critical circuitry). Reverse engineering techniques, such as extracting program code and disassembly/debugging methods, are simplified greatly in a software environment, allowing a token's secret components such as cryptographic algorithms, private keys, and other assumed secure information to be recovered.

Due to portable devices (such as Palm and PocketPC) becoming commonplace in the industry [6], vendors of hardware tokens are developing software-based versions for use on such devices. Software token applications also exist on desktop systems which usually have network connectivity (allowing for additional attack vectors). Regardless of the fact that these devices can now provide access to critical corporate assets, they are often left unattended and unprotected.

3.1 Hardware v. Software Trade-offs

Software tokens provide convenience because they operate on a platform that the user already has access to, such as a laptop or PDA. They do not require owning an application-specific piece of hardware and do not add another piece of equipment that could be lost or stolen. Software tokens allow the execution of an application on a previously secure device to be embodied on an insecure platform. Methods to determine program operation are much easier in this fashion, making the software tokens less secure and causing a weak link in the security chain.

Hardware token devices often contain tamper detection mechanisms that software-based devices lack, and will sometimes erase critical information from the device if physical tampering is detected [3] [4] [11]. This is not to say that dedicated hardware-based devices are immune to attack and do not contain security design flaws [1] [14]. However, the difficulty and required cost to attack is increased.

The availability of free and commercial decompilers for software token environments such as Windows, Palm OS, and Java makes the software reverse engineering task a more likely one than hardware counterparts. Most software-based token devices have been publicly available for a relatively short period of time. It is likely that, given the known insecurities and problems with software devices, attacks on the software-based authentication technologies will become more commonplace.

Tables 1 and 2, consisting of a small sampling of the available authentication token devices, show that software-based devices currently lay claim to most of the security-related flaws compared to dedicated hardware devices.[2]

The primary and often easily provable concern for software tokens is the possibility of extracting secret component and PIN information from a legitimate token, which can lead to a complete cloning of the device by an attacker. The cloning of software-based tokens has been demonstrated with two products [7] [8]. The extraction of a token's proprietary tokencode generation algorithm has also been achieved [15]. Once the cryptographic algorithm of the authentication device has been determined, further analysis of the product can take place [10]. Possible flaws in the algorithms may uncover repeating sequences of tokencodes or the capability to determine secret components or future tokencodes by viewing previously generated tokencodes.

Although it is difficult to completely protect credentials and algorithms stored on a software-based device, there are always ways to improve the security. It is recommended that vendors properly encrypt and salt credentials if they must be stored on the device. Simple obfuscation and transforms that can be reversed or brute-forced lull the user into a false sense of security and show a lack of concern about security from the vendor. The use of a salt minimizes the possibilities of a password being represented in the same fashion on multiple systems. A method such as "cryptographic camouflage" could also be implemented [5].

[2]These tables contain a selection of authentication token device vendors used simply for demonstrative purposes. It is not a comprehensive listing of all one-time password token vendors. Other such vendors include VASCO, ActivCard, and Symantec.

In a software environment, the application inherits the same level of security as the operating system it is running on. Regardless of how data may be salted or encrypted, ultimately it must be decrypted for processing and stored in plaintext in device memory (which may be accessed by an attacker). There are too many factors and uncontrollable circumstances of the software environment to place trust in a software-based token device.

Vendor	**Token Name**	**Released**[3]	**Publicly Known Security Issues**
CRYPTOCard	RB-1	1994	None
	KT-1	Nov 1999	None
RSA Security	SecurID	1986	None
Secure Computing	SafeWord Platinum	pre-1992	None
	SafeWord Silver 2000	N/A	None
	SafeWord Gold 3000	2001	None

Table 1: Hardware Tokens: Release Dates and Security-Related Public Announcements

Vendor	**Token Name**	**Released**[3]	**Publicly Known Security Issues**
CRYPTOCard	ST-1 Java	Nov 1997	None
	PT-1 Palm OS v1.04	Nov 1999	Can clone token, Apr 2000 [7]
RSA Security	SoftID	1996	Algorithm released, Dec 2000 [15]
			Potential weaknesses, Jan 2001 [10]
	SecurID Palm OS	Apr 1999	None
	SecurID Nokia 9210	Dec 2000	None
	SecurID Ericsson R380	Nov 2000	None
Secure Computing	SofToken PC	Feb 1995	None
	SofToken Kyocera pdQ	May 2000	None
	e.iD Palm OS v2.0	Mar 1999	Can clone token, Dec 2000 [8]
	e.iD Windows CE	N/A	None
	e.iD Ericsson R380	Sep 2000	None

Table 2: Software Tokens: Release Dates and Security-Related Public Announcements

[3]The data refers to the initial release version of the token product unless otherwise noted. Release dates and version numbers are shown to distinguish the listed products from future products and updates. This allows vendors to attach appropriate version numbers to updated products, if they exist, in order to avoid end-user confusion.

4 Conclusions

The CERT Coordination Center [2], the Common Vulnerabilities and Exposures Project [9], and the BUGTRAQ Vulnerability Database [13] show the large number of attacks on software and software-based technologies. Hardware attacks are less common. However, administrators and users should not be complacent simply because they implement hardware-based authentication devices instead of software-based ones.

The educated selection of an authentication token technology depends on a number of factors, including risk management, business needs, TCO issues, and security. No single technology will provide the ultimate solution for every situation, and there are advantages and disadvantages to the use of each type. Care needs to be taken to ensure that the product is carefully analyzed before it is deployed in a particular organization's infrastructure.

References

[1] R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," *Security Protocols, 5th International Workshop*, 1997.

[2] CERT Coordination Center, "CERT/CC Statistics 1988-2001," `http://www.cert.org/stats/cert_stats.html`

[3] D. Chaum, "Design Concepts for Tamper Responding Systems," *Advances in Cryptology: Proceedings of Crypto '83*, 1984.

[4] A. J. Clark, "Physical Protection of Cryptographic Devices," *Advances in Cryptology: EUROCRYPT '87*, 1988.

[5] D. N. Hoover and B. N. Kausik, "Software Smart Cards via Cryptographic Camouflage," *IEEE Symposium on Security and Privacy*, 1999.

[6] IDC, "Market Mayhem: The Smart Handheld Devices Market Forecast and Analysis, 1999-2004," Report 22430, June, 2000.

[7] Kingpin and DilDog, "CRYPTOCard PalmToken PIN Extraction," *@stake Security Advisory*, April 10, 2000, `http://www.atstake.com/research/advisories/2000/cc-pinextract.txt`.

[8] Kingpin, “SafeWord e.iD Palm Authenticator PIN Extraction,” *@stake Security Advisory*, December 14, 2000, `http://www.atstake.com/research/advisories/2000/a121400-1.txt.`

[9] The MITRE Corporation, “Common Vulnerabilities and Exposures,” `http://cve.mitre.org/cve/index.html`

[10] Mudge and Kingpin, “Initial Cryptanalysis of the RSA SecurID Algorithm,” January 2001, `http://www.atstake.com/research/reports/initial_securid_analysis.pdf.`

[11] National Institute of Standards and Technology, “Security Requirements for Cryptographic Modules,” *FIPS 140-1*, January 1994, `http://www.itl.nist.gov/fipspubs/fip140-1.htm.`

[12] B. Schneier, “Secrets & Lies,” Chapter 19: Threat Modeling and Risk Assessment, John Wiley & Sons, 2000.

[13] SecurityFocus.com, “BUGTRAQ Vulnerability Database Statistics,” `http://www.securityfocus.com/vdb/stats.html`

[14] S. H. Weingart, “Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses,” *Workshop on Cryptographic Hardware and Embedded Systems*, 2000.

[15] I. C. Wiener, *Sample SecurID Token Emulator with Token Secret Import*, BugTraq posting, December 21, 2000, `http://www.securityfocus.com/archive/1/152525`.