USENIX Association

Proceedings of the 11th USENIX Security Symposium

San Francisco, California, USA
August 5-9, 2002

VeriSign CZAG: Privacy Leak in X.509 Certificates

Scott G. Renfro
Yahoo!, Inc. *
scott@renfro.org

Abstract

We first analyze a concrete example of embedding sensitive information in X.509 certificates: VeriSign's CZAG extension. Second, we consider the general case of a sharing certified information with a mutable subset of relying parties. The example nicely illustrates several well-known technical, social, and economic issues through the effective publication of users' country, zip code, date of birth, and gender in as many as three million certificates over a five year period ending in 2002. The general case continues to arise in many new PKI deployments, where system designers are pressured to include potentially sensitive information in end entity certificates. Ultimately, failure to carefully consider the risks when developing a certificate profile may allow sensitive information to leak outside the intended scope.

1 Introduction

In recent years, Ellison [7, 9], Brands [3], and other authors have warned against embedding personal and authorization data in identity certificates, noting the difficultly of controlling disclosure of such data. Despite such warnings, many system designers have been lured by the convenience of including a wide range of subject attributes, primarily through the extension mechanism available in X.509v3.

Some of this information is quite sensitive and should not be disclosed except to a few trusted parties, while much is as public as the subject's distinguished name. In many cases, however, the information falls somewhere in between — it may be public or widespread knowledge, yet binding it inside the certificate risks building a convenient dossier that the user presents to anyone who requests it.

In 1997, VeriSign introduced such a feature in their Class 1 certificates. These certificates are intended to bind a user-generated public key to an e-mail address for use with S/MIME or SSL. This feature, informally called CZAG (Country, Zip, Age, Gender), included subscriber–entered demographic information, when submitted at registration time. While users may opt out during registration, most subscribers supplied their personal data. On registration forms and documentation, , this feature was called the One-Step Registration field, since it promised to enable registration at participating web sites in one easy step.

When provided, the CZAG information was stored encrypted in the subscriber's certificate and could be read by participating web sites using software available from VeriSign for a licensing fee. Unfortunately, this software was not necessary to read the data and anyone could read the certificates from VeriSign's public LDAP server.

Several factors lead users — even industry insiders who failed to opt–out — to believe their information was reasonably protected. First, the data was encrypted and couldn't be seen by visual inspection. Second, press releases and the registration web site explained that the demographics would only be made available to trusted sites who have signed a strict license agreement, and that misuse of the data would cause revocation. On the other hand, VeriSign never explicitly claimed to protect or "encrypt" the data, and their privacy statement explicitly states that users should have no expectation of privacy regarding any data included in certificates.

Regardless of the disclaimers, it is unlikely that most subscribers recognized that their personal demographic information was so significantly exposed. Since VeriSign publishes all Class 1 certificates in their public directory server and the information was only protected by trivially weak encryption, subscriber personal demographic information was effectively published for all to read.

In the first half of this paper, we examine VeriSign's CZAG extension in greater detail, providing some his-

*Worked performed at Securify, Inc.

tory, details on the data encoding and encryption (simply XOR'd with a constant), and a demographic summary of more than 16,000 certificates. This example nicely illustrates some of the technical, social, and economic issues that interact in real-world situations.

In the second half, we describe the more general problem of a Certification Authority sharing sensitive data with a changing subset of relying parties, including the constraints imposed by the nature of X.509 public key infrastructures. We highlight alternate design approaches that may lead to better solutions than those used in the example. This general case continues to arise in new PKI deployments, as system designers are asked to include potentially sensitive data in certificates.

2 VeriSign's CZAG Extension

VeriSign's CZAG extension illustrates several well-known technical, social, and economic issues, which we discuss further below. First, the system used unexpectedly weak encryption and had no revocation mechanism. Second, there was a clear discrepancy between users' expectations and the actual protection promised and delivered. Finally, there was no economic motivation to correct the known weaknesses — despite affecting millions of users over several years, the lack of licensees meant that nobody was paying for the privilege of accessing the information.

2.1 History

In 1997, VeriSign announced an optional One-Step Registration feature that included a user's country, zip code, date of birth, and gender in Class 1 certificates when the users do not opt-out[16]. Although subscriber's are clearly advised that the information is optional, inclusion is the default and most subscribers provide the information (see § 5). As part of the announcement, VeriSign described the availability of an implementation kit available for an annual licensing fee. This kit includes a registration license key to read the One-Step Registration field[16].

To subscribers, the One-Step Registration field was marketed as a way to ease web site registration, promising to bring personalized content and eliminate repetitive data entry. To web sites, however, the feature was always marketed as a way to anonymously track the demographic make-up of people visiting their sites[17]. In practice, many users entered their actual demographics and real names, leaving little anonymity.[1]

The addition of this information to VeriSign certificates generated some discussion within the PKI community and further fueled the debate on appropriate uses of X.509v3 certificates — especially the privacy issues involved in binding a public key and other attributes to the X.500 concept of a unique identity[12].

2.2 Technical

Unfortunately, the VeriSign CZAG feature suffered from at least two weaknesses that we discuss further below. The demographics were only weakly masked and there was no technical mechanism to revoke sites whose contract lapsed or was terminated for misuse.

2.2.1 Trivially Weak Encryption

The encrypted demographic information included in VeriSign Class 1 certificates was encoded into a private X.509v3 extension. When generating the certificates, the demographic information was first written into fixed–length fields within a larger data structure. This data structure was then encrypted, base–16 encoded (i.e. a hex string), DER[2] encoded as an IA5String, and finally enclosed within the octet string of a private X.509v3 extension. This extension was included in the user's certificate.

We next examine how to decode, decrypt, and interpret the extension contents. To ensure our description is accessible to most readers, we include details that readers familiar with X.509, ASN.1[3], and DER will already understand and should feel free to skip over.

The encrypted demographic information in VeriSign Class 1 certificates is conveyed in an X.509v3 extension, the ASN.1 syntax of which is shown in Figure 1. In layman's terms, this syntax means that the extension is uniquely identified by the `id-verisign-czag` object identifier, and the contents included in the extension's payload (which is encoded as an octet string) con-

[1] We even saw cases where users with pseudonymous e-mail addresses, provided a verifiable full name, date of birth, and zip code (e.g., hackerd00d@example.com).

[2] DER stands for ASN.1 Data Encoding Rules.

[3] ASN.1 stands for Abstract Syntax Notation

sist solely of an IA5String, a common ASN.1 string type.

```
id-verisign-czag OBJECT IDENTIFIER ::=
  { 2 16 840 1 113733 1 6 3}

verisignCZAG EXTENSION ::= {
  SYNTAX VerisignCZAGExtension
  IDENTIFIED BY id-verisign-czag }

  VerisignCZAGExtension ::= IA5String
```

Figure 1: ASN.1 syntax for the CZAG extension

According to DER, the `id-verisign-czag` object ID is encoded as the byte stream `060a6086480186f845010603`. One can simply check a certificate for presence of this byte stream to determine whether it contains the CZAG extension.[4]

The 116 hex characters within the IA5String decode to 58 bytes of obviously structured ciphertext. Within the thousands of certificates we examined, only 19 of the 58 bytes ever varied; the remaining 39 bytes were constant. Further, the distribution of the variable bytes of ciphertext was not uniform, but limited to a small range of values. These characteristics are not typical of strongly encrypted data.

A chosen plaintext attack — mounted by registering for several free e-mail certificates — quickly reveals that the data is encrypted with an unknown stream cipher using a single fixed key. Given a handful of samples, we recovered the portion of keystream corresponding to the 19 variable bytes of ciphertext. The effect is functionally equivalent to an every-time-pad (i.e., the same keystream is used to encrypt the CZAG extension in every VeriSign Class 1 certificate examined, unlike a one-time pad where the keystream is randomly chosen and never reused).

It isn't clear if VeriSign simply made implementation errors when integrating a relatively trusted stream cipher (e.g., RC4) or if the original implementation only masked via XOR. The presence of additional, constant bytes suggests that some portion — perhaps the first 32 bytes (256 bits) — correspond to either an encrypted, per–message key or a key identifier. If this is the case, however, the key never changed and the same keystream resulted for every certificate.

The recovered keystream k shown in Figure 2 represents each fixed byte as `00`. The plaintext of the demographic structure can then be easily recovered since $p = c \oplus k$ where $\oplus$ represents the byte–wise XOR operation.

```
k = 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000
    0086a100 0000fb0b f2c8b226 9d5bc1e7
    0079ae93 8b72cd00 a700
```

Figure 2: Recovered keystream

After decryption, the demographic data can be simply decoded into its components. There are four pieces of information in the structure: country, zip/postal code, date of birth, and gender. The format of the plaintext is most easily expressed by the ANSI C structure shown in Figure 3.

```
struct demographics {
  char unknown1[33];
  char country[2];
  char unknown2[3];
  char zipcode[10];
  char unknown3;
  char dob[6];
  char unknown4;
  char gender;
  char unknown5;
};
```

Figure 3: ANSI C structure showing layout of field

Within this structure, country is represented as a two–byte country code contained in the 34th and 35th byte of the One-Step Registration field. The country code corresponds to countries according to Table 1.

Similarly, the zip code is represented as the ten ASCII characters entered by the subscriber. The contents are left–padded with spaces (i.e. when the user enters less than ten characters during registration, spaces are inserted from the left until the string totals ten characters). The ten characters are located in bytes 39 to 48 of the plaintext.

The date of birth entered by the subscriber is represented as six characters from bytes 50 to 55 of the plaintext. The characters are formatted `MMDDYY` so that February 01, 1970 appears as `020170`.

Finally, gender is simply one character — either `M` for Male or `F` for Female. This character is byte 57 in the plaintext. We didn't find any certificates where this byte decoded to some other value of gender.

[4] There is a very small possibility of a false positive if the sequence appears in a public key, signature, or another extension's payload.

Code	Country	Code	Country
AU	Australia	JP	Japan
AT	Austria	MX	Mexico
BE	Belgium	NL	Netherlands
BR	Brazil	NO	Norway
CA	Canada	ZA	South Africa
CN	China	ES	Spain
DK	Denmark	SE	Sweden
FI	Finland	CH	Switzerland
FR	France	TW	Taiwan
DE	Germany	GB	United Kingdom
IN	India	US	United States
IL	Israel	UU	Other Countries
IT	Italy		

Table 1: Country codes used in the One-Step Registration field

2.3 Survey of Demographics

Using the preceding information, sample certificates were retrieved from VeriSign's public directory server at `ldap://directory.verisign.com`[5] and analyzed to verify the decryption and decoding algorithms, while collecting some broad statistics.

Of the certificates examined, 77% included the CZAG extension. Summarized demographics of those certificates are listed in Table 2. These statistics are based on a non-random sample of 16,285 certificates after removing those with either missing data, age less than 10 years, or age greater than 80 years. Certificates with ages greater than 80 and less than ten nearly always resulted from intentional or inadvertent data entry errors (e.g. listing the current year in the subscriber's date of birth).

We also estimated the total number of certificates issued based on numbers VeriSign has made public directly or through news reports. Prior to the CZAG extension being introduced, VeriSign had issued more than 750,000 consumer end–entity certificates (excluding SSL server certificates) [14]. VeriSign passed the 3 million client certificate mark sometime between October, 1998 [5] and June, 1999 [4], not including OnSite certificates for corporate users. This allows us to conservatively estimate that VeriSign issued more than 5 million client certificates during the time that CZAG extensions were being embedded. Given the 77% inclusion rate, we deduce that more than 3 million certificates issued between 1997 and 2002 may have included the CZAG extension.

[5] A user's base–64 encoded certificate can be simply retrieved with the command line: `ldapsearch -h directory.verisign.com -b "" mail=<email> usercertificate;binary`

2.3.1 Access Revocation

From the preceding discussion and our examination of thousands of certificates over several years, it is clear that only a single encryption key was used to encrypt this data. As a result, VeriSign had no way to revoke access to the data by a trusted third party whose agreement had either expired or been terminated.

Although the initial press releases [15] claimed that a third party who violated the privacy pledge would have their license revoked, there is no obvious technical mechanism to enforce this action. The CZAG extension was simply to small to contain separately encrypted message encryption keys for each trusted third party. As a result, VeriSign would have had to revoke access to all parties by changing the symmetric encryption key, only distributing the new key to those parties that were still trusted.

Finally, such an approach didn't prevent these now–untrusted third parties from accessing the data within older certificates, only within the certificates encrypted with the new key. Since the certificates are issued with a one-year validity period, it may take up to one year for these old certificates to fall out of circulation.

Category	Male		Female		Total	
Age 10 – 18	162	(1%)	23	(<1%)	185	(1%)
Age 18 – 24	785	(5%)	156	(1%)	941	(6%)
Age 25 – 34	2,992	(18%)	577	(4%)	3,569	(22%)
Age 35 – 45	4,065	(25%)	746	(5%)	4,811	(30%)
Age 45 – 55	3,796	(23%)	648	(4%)	4,444	(27%)
Age 55 – 65	1,569	(10%)	204	(1%)	1,773	(11%)
Age 65 – 80	495	(3%)	67	(<1%)	5,62	(3%)
Total	13,864	(85%)	2,421	(15%)	16,285	(100%)

Table 2: Demographic summary of examined certificates

2.4 Social

In addition to technical issues, the VeriSign example illustrates a common social/policy issue: there was a clear discrepancy between user expectations based on marketing literature and VeriSign's actual commitment and implementation.

Subscribers, including a few industry experts to whom we sent birthday greetings during our original research, incorrectly believed that their information was only available to web sites licensed by VeriSign. Because the information was encrypted, it was not visibly exposed when inspecting the certificate in a web browser or with ASN.1 parsers.

Additionally, most of the public documentation implies limited distribution and availability of the user's demographics when provided — even to the technically knowledgeable reader. The registration page indicates the data is "presented to participating web sites"[19], while a press release[16] claims "consumers have complete control of . . . whether or not to present it at sites they visit." Finally, VeriSign's original announcement asserted that participating sites must adhere to a consumer privacy pledge[16] and sites which violate the provisions of the pledge will have their reader license revoked by VeriSign[15].

Although much of this documentation implies limited distribution and strong protection, VeriSign never explicitly asserted such protection. In fact, the word "encryption" was never used in conjunction with the feature and their privacy statement points out that "VeriSign's CPS requires VeriSign to publish all subscriber certificates within the Public Certification Services. Consequently, a subscriber should have no expectation of privacy regarding the content of his or her Digital ID."[18]

This conflict is common among on-line services. They have a strong economic motivation to build trust among users, yet cannot explicitly misrepresent data handling and disclosure procedures. As a result, most public descriptions euphemistically describe the features positive aspects and gloss over any risks. Finally, the two types of text — legal and marketing – are typically authored by different people with somewhat opposing motivations, increasing the confusion.

This situation has only worsened over the last several years as economic stakes increase and user expertise drops. Many hope that the Platform for Privacy Policies Project (P3P) will yield a useful and constrained language used to communicate these privacy practices to users, and allow users to easily program their user agents to automatically take action based on a site's machine–readable policies[6]. Ultimately, however, users who want to control information on a per–attribute basis may find P3P wanting.

2.5 Economical

The third interesting characteristic of the VeriSign example has been it's resistance to change. Although it was rolled out more than five years ago with known weaknesses and was never a significant source of revenue, VeriSign was been reluctant to remove the feature from their registration process.

When launched in 1997, VeriSign announced that a handful of companies had already agreed to license the software and use the data in their registration process. These companies hoped to get accurate demographics from users who were registering for services. There was also an expectation that users would be more willing to register at new sites since the information required by the service provider was transfered automatically. Finally, this was a new application for the use of client–side certificates. Ideally, the lure of one password and one–step

registration would encourage users to pay VeriSign to get such a convenience at the same time sites were paying for access to the data.

Unfortunately for VeriSign, and much of the PKI industry, the use of client–side SSL certificates never really caught on. There were several problems, including software compatibility, mobility, and lack of motivation. Since so few sites supported the CZAG extension or client authentication SSL, the certificates were relegated to use in secure e-mail applications, and the initial sites dropped support.

In January 2000, when we first examined the CZAG extension, we provided an early predecessor of this paper to VeriSign, and later other members of the information security community. In response, VeriSign said that the problems were old news, the extension had been long forgotten by sites, and concerned users could always opt-out. Although true, thousands of users registering for new certificates each month were still supplying demographics only to have them published in VeriSign's public LDAP directory.

In retrospect, their response was not surprising. As Anderson points out [1], the real driving forces behind information security issues commonly have more to do with perverse economic incentives than technical issues. When a party chooses to leave a potential security problem in place rather than correct it, it may well be the result of their cost–benefit analysis.

In this case, having the feature in place cost VeriSign nothing while removing the feature incurred cost and risk. Since the initial sites had all dropped support, nobody was paying for the privilege of accessing the data. Thus there was nobody to complain that others, who hadn't paid a licensing fee, also had access. Despite any theoretical cost borne by unsuspecting users, it made economic sense to delay removal until there was independent cause to do so.

3 The General Case

The VeriSign CZAG extension is just one example of a relatively common problem in X.509 deployments: how can a Certification Authority share sensitive information with some subset of relying parties without unnecessarily disclosing that information to unauthorized parties.

This problem arises in a surprising number of PKI deployments. For example, "Qualified Certificates" are designed for use in legally binding contexts and may contain a government–issued unmistakable identifier (e.g., social security or drivers license number)[13]. Yet broad publication of this identifier, bound to the associated subject name, may lower the barrier to identity theft. Other examples include patient identifiers, professional license numbers, and internal corporate usernames, hostnames, or IP addresses. Even embedded authorization data may disclose information that is useful to both attackers and users.

As a result, many system designers struggle with whether to include such information and, if included, how to protect it. In the following sections, we outline goals and design constraints that should be considered and then suggest some directions that may lead to appropriate solutions. We discuss embedding opaque identifiers that point to an online database, protecting the data with various key management schemes, and tools to allow the user to control disclosure.

We don't further consider the obvious case of simply accepting the risk posed by embedding sensitive information in certificates without further protection.

3.1 Goals

Although we implicitly touched upon many of the goals while discussing the CZAG example, it is useful to state these explicitly. Our overall objective is to make certified information available to, and only to, a mutable subset of relying parties. We call members of this subset "trusted relying parties."

- Protect the confidentiality and integrity of data in storage and transmission, between the time the user submits it and a trusted relying party accesses it.
- Prevent unauthorized parties from accessing the data at any time.
- Allow strong access revocation for relying parties that were once authorized but are no longer. Ideally, once revoked, they will be unable to access any sensitive information. Practically, preventing access to information not already in their possession may be sufficient.
- Trusted relying parties should be able to act independently. Changes to the set of trusted relying parties should only affect those parties whose status

has changed and not parties whose status has not changed (e.g., when one party's access is revoked, there should be no effect on other, still–authorized parties).

3.2 Design Constraints

The format of X.509 certificates and deployment model characteristics (e.g., on–line vs. off-line systems) combine to impose several design constraints on the problem. These constraints eliminate many of the traditional solutions to similar problems.

- Any data added to a certificate must be relatively small in size. Some applications impose a 2 kilobyte limit on the overall certificate size, which limits the extension to about 1kb.
- As a result, any protection must not suffer from significant message expansion (e.g., a S/MIME blob would almost always exceed the size constraint).
- Certificates remain static for relatively long periods of time (typically one year). Revocation and reissuance is a messy proposition at best.
- In most applications, not only may the certificate and its contents become public, they are assumed to be public knowledge.

Designers must add their own goals and constraints to this baseline. For example, real–time communication with the Certification Authority may not be possible in off–line systems. Some on-line systems may have performance requirements that preclude additional network transactions during certificate validation. Finally, some systems may have legal restrictions imposed on the strength of the protection employed. Most systems being deployed today, however, are database–centric and on–line, performing network transactions during certificate validation (e.g., Online Certificate Status Protocol checks).

3.3 Online Database

The most obvious solution to this problem is to not embed the sensitive data in the public certificate at all. Instead, the Certification Authority stores sensitive information in a centralized database. This information can be indexed by any of several values embedded in the certificate, including an opaque identifier, the subject distinguished name, or the issuer name, serial number pair (which is required to be unique within X.509).

Trusted relying parties can query the database with their own credentials and the index of the information desired. Assuming they are authorized, the information is returned. Their interface to this database may be LDAP, a relational database protocol, or a custom application protocol.

If additional protection against data modification while in storage is desired, the Certification Authority can use issue–time binding. One approach is to append a nonce to the information in the database and including the hash of the information and nonce in the end–entity certificate. The nonce prevents identical information from hashing to the same result. This allows the relying party to verify that the information has not changed since the certificate was generated. On the other hand, if the information needs frequent updates, issue–time binding may not be appropriate.

The primary drawbacks to a database-centric solution are latency and the risk posed by a central database. Additionally, such a scheme is not practical for off–line systems unless they can batch requests for later analysis.

In return, the system can ensure that only authorized parties have access to the database and not expose ciphertexts to untrusted parties. Further, access revocation can be nearly instantaneous and totally independent. In this case, once a party's access has been revoked, they lose access to all data that they have not previously stored within their own systems. Finally, a database can allow finer grained access control and greater flexibility under changing requirements.

3.4 Key Management

When the application requires that information be embedded in certificates (e.g., because it is off–line), better key management schemes can improve the security of this practice.

First, the data should be better protected against cryptanalysis. One option is to use a block cipher with a random, per–certificate IV (i.e.,

$$M = IV, E_{k_m}(data)$$

where M is the resulting certificate extension, IV is the

random, per-certificate initialization vector, $data$ is the sensitive data and k_m is the master encryption key).

Alternatively, it is possible to encrypt the data with either a stream or block cipher using a per–certificate key derived from a single master key combined with unique information in the certificate (e.g., the public key or subject distinguished name). In this case,

$$k_c = f(certificate, k_m)$$

and

$$M = k_c(data)$$

where k_c is a per-certificate key derived from the master key material and the certificate.

Such simple approaches ensure that only those with proper key material can decrypt the data and are sufficient in a closed system where the Certification Authority is the only relying party.[6] In all other systems, revocation remains an issue.

3.4.1 Key Rotation

When the subset of trusted relying parties is small or only changes infrequently, it may be sufficient to have a single master key which is changed according to some fixed schedule. This allows a tradeoff between administrative overhead and revocation speed.[7]

In general, administrative overhead is directly related to frequency of key rotation. Worst–case revocation speed can be derived from key lifetime, T_k, certificate lifetime, T_c, and key distribution lead time[8], T_d. Loss of access to new data begins to take effect no later than $T_k + T_d$ after revocation and requires an additional T_c to complete.

For example, assume we generate certificates with one–year validity periods, change keys annually, and distribute keys one month before they are required. This approach causes very little additional overhead (just annual updates), but revocation speed is quite slow, with revoked parties unaffected for up to fourteen months and retaining access to some data for up to twenty–six months after revocation.[9] Similarly, when issuing six-month certificates with monthly key rotation and one-month key distribution lead time, revocation begins to take effect after two months and leaves revoked parties completely without access no more than six months later. However, this comes at a substantially greater cost in administrative overhead.

One variation is to trade scheduled key rotation for reactive key rotation that only occurs when there has been a revocation. Each key rotation is typically more expensive since it was unplanned, but there may be fewer overall key changes as a result. In this case, revocation begins to take effect after just T_d and requires another T_c to complete. Unfortunately, each revocation of a single party requires all parties to change keys – a tremendous administrative burden.

These approaches cause little message expansion, but significant administrative overhead. Additionally, they leave revoked parties with complete access for a significant period of time or penalize all parties when a single party is revoked. Fortunately, it is possible to reduce administrative overhead while improving revocation speed.

3.4.2 Group Keys

As the subset of trusted relying parties grows or changes more frequently, it may be more efficient and more effective to randomly split the trusted parties into key groups. All parties within a single key group share a common key encryption key.

The data is first encrypted using a randomly–generated, per–certificate data encryption key and that key is encrypted with each group's key encryption key. This potentially includes some spare keys which are not initially distributed to any parties. The resulting message is composed as

$$M = E_{k_1}(k_c), E_{k_2}(k_c), \ldots, E_{k_{n+s}}(k_c), E_{k_c}(data)$$

where k_c is the per–certificate data encryption key, n is the initial number of key groups, s is the number of spare group keys not initially distributed, and k_1 through k_{n+s} are the group key encryption keys.

When a party's access is revoked, their group's key is removed from use (in future messages their key is used

[6] Such systems are surprisingly common amongst closed public key infrastructures.

[7] Revocation speed is the time required for a revocation event to result in loss of access for the revoked party.

[8] Key distribution lead time is how long prior to using a particular key we distribute it to trusted parties.

[9] For example, consider a party who becomes unauthorized after the following year's key has been distributed but before it has been put into use (e.g., Dec 15, 2001). A certificate issued under that key on the last day of the key's use (e.g., Dec 31, 2002) will remain valid and in circulation through Dec 30, 2003 (for a one–year validity period) and the now–unauthorized party will have had access on a date for just over two years after their authorization was revoked.

to encrypt a fixed, worthless value rather than the actual data encryption key). The remaining authorized parties within that group are given one of the spare keys or, if no spare keys remain, another group's key. This effectively merges them into a new group.

Compared to simple key rotation, group keys reduce administrative overhead and increase independence of the parties by localizing the effect of a revocation. Additionally, worst–case revocation speed matches the best–case when using key rotation alone. In the worst–case, a revoked party begins to lose access to some data after T_d and has lost all access after $T_c + T_d$.

The cost of group key management is primarily borne in message expansion and size constraints limit us to a relatively small number of groups. Generally speaking, the maximum number of group keys, $n + s$, given a total space of S_t, a plaintext message of size S_m, and a key of size S_k is $n + s = \lfloor (S_t - S_m)/S_k \rfloor$. For example, a 250 byte message within a 1000 byte space would allow for 46 independent 128-bit keys.

Much more sophisticated approaches are used in the copy protection [11], and broadcast and multicast key management [21, 20] fields.

3.5 Other Solutions

Finally, it may make sense to depart from the X.509 identity certificate approach completely, putting control of the information in the user's hands.

One option, though not widely supported, is to issue X.509 attribute certificates for the user's attributes. These certificates typically bind subject attributes to an entity, rather than an entity to a public key [10]. Used in conjunction with identity certificates, they would allow users to authenticate with their identity certificate and then present the appropriate attribute certificate as necessary. Often, there may be one attribute certificate that contains all attributes. A more flexible approach is to generate one certificate per attribute. This allows the user to choose which attributes to disclose and which to hold private. Ease of use would depend on application support for managing groups of related attributes. X.509, however, expects the user to first authenticate their identity and then authenticate their attributes, but not the latter without the former.

One variation on this concept simply binds attributes to a public key and foregoes the identity concept all together. This allows the user to present the truly required data (e.g., "Is the presenter authorized?" or "Do their attributes meet certain criteria?") without actually forcing them to reveal their identity. Brands developed an early proposal of this idea in 1993 [2] based on many of the themes in Chaum's work. This same concept is central to the Simple Public Key Infrastructure (SPKI) standards developed by Ellison and others [8].

These approaches give control to the user, who has the economic incentives to exert granular control over the information.

4 Conclusion

The inclusion of the CZAG extension in subscriber certificates is representative of a more widespread temptation to use X.509v3 certificates as a carrier for many kinds of subject attributes not needed for the actual purpose of the certificate. Although subscribers had the opportunity to opt–out of the feature, most did not. It is unlikely that many of these users realized their personal data was not just available to a few participating web sites, but was published on the Internet where it was readable by anyone given trivial effort.

Organizations planning and deploying both open and closed public key infrastructures are frequently expected to embed sensitive or potentially sensitive information in user certificates. In the face of such requirements, designers must carefully consider the risks when developing an appropriate certificate profile. Failure to do so may allow private data to leak outside the intended scope.

5 Acknowledgments

We sincerely appreciate the efforts of everyone who provided feedback on early predecessors of this paper, including Taher Elgamal, Mark Chen, and Mark Schertler. The anonymous reviewers' comments were helpful in developing the paper's final structure. Many thanks, also, to Peter Gutmann who originally recommended polishing up the paper for submission and whose `dumpasn1` tool is incredibly handy.

References

[1] Ross Anderson. Why Information Security is Hard — An Economic Perspective. *Annual Computer Security Applications Conference*, December 2001. http://www.acsac.org/2001/papers/110.pdf.

[2] Stefan Brands. Privacy–protected transfer of electronic information, 1993. United States Patent.

[3] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates — Building in Privacy*. Stefan Brands, 1999.

[4] James Brandt. Children's Online Privacy Protection Rule — Comment P994504. http://www.ftc.gov/privacy/comments/verisigninc.htm, June 1999.

[5] Tim Clark. Verisign adds key recovery. *news.com*, October 1998. http://news.com.com/2102-1017-216571.html.

[6] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. TR P3P, W3C, April 2002. http://www.w3.org/TR/P3P/.

[7] Carl Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI Certificate Theory. RFC 2693, IETF, September 1999. http://www.ietf.org/rfc/rfc2693.txt.

[8] Carl M. Ellison. Establishing identity without certification authorities. *Proceedings of the Sixth Annual USENIX Security Symposium*, pages 67–76, 1996. http://citeseer.nj.nec.com/ellison96establishing.html.

[9] Carl M. Ellison. Naming and certificates. *Computers, Freedom, and Privacy*, April 2000.

[10] ITU. The Directory—Authentication Framework. *ITU Recommendation X.509*, 1997.

[11] Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. In *CRYPTO*, pages 41–62, 2001.

[12] ColorNet Robin, Carl Ellison, and Ed Gerck. re: [E-CARM] Certificates and privacy and VeriSign. http://www.mail-archive.com/cert-talk@structuredarts.com/mail3.html, May 1998.

[13] Stefan Santesson, Tim Polk, Petra Barzin, and Magnus Nystrom. Qualified certificates profile. RFC 3039, IETF, January 2001. http://www.ietf.org/rfc/rfc3031.txt.

[14] VeriSign. Leading web sites accept verisign digital ids. http://corporate.verisign.com/news/1997/sites.html, April 1997.

[15] VeriSign. Verisign and etrust team up to assure consumer privacy on the internet. http://www.verisign.com/press/partner/etrust.html, April 1997.

[16] VeriSign. Verisign enhances digital ids to enable universal website login and one-step registration. http://www.verisign.com/press/product/isv.html, April 1997.

[17] VeriSign. Verisign digital id usage. http://www.verisign.com/about/id_imp.html, 1998.

[18] VeriSign. Verisign, inc.'s overall privacy statement. http://www.verisign.com/truste/, February 1998.

[19] VeriSign. Microsoft class 1 enrollment. https://digitalid.verisign.com/client/class1MS.htm, 1999.

[20] Debby Wallner, Eric J. Harder, and Ryan C. Agee. Key management for multicast: Issues and architectures. RFC 2627, IETF, June 1999. http://www.ietf.org/rfc/rfc2627.txt.

[21] Wong, Gouda, and Lam. Secure Group Communications Using Key Graphs. *IEEETNWKG: IEEE/ACM Transactions on Networking*, 8, 2000. http://citeseer.nj.nec.com/wong98secure.html.