The Authenticated Grade Report System

Kimberly Kuo, Sha Ma and Michael Salib

May 9, 2002

Abstract

Inefficiencies of paper transcripts have spurred the development of digital transcripts. We describe one such system, the Authenticated Grade Report (AGR) system, which allows universities to generate and verify grade reports, transcripts and diplomas using public key cryptography. The system relies on the US Department of Education to act as a central clearinghouse for university public keys and key revocation lists. Individual professors give each student an AGR while universities issue Authenticated Transcripts and Diplomas (ATD). Any party can verify the authenticity of an AGR or ATD by contacting the USDE in order to receive the signing university's public key and the latest key revocation lists, even if the signing university has disintegrated. The system is designed to be simple, explicit and open.

Contents

List of Figures

1 Introduction

A secure system should provide authentication, authorization and confidentiality. In this paper, we discuss the design of a secure system which generates and distributes cryptographically authenticated grade records. At the end of each semester, instructors will cryptographically sign and give each of their students an Authenticated Grade Record (AGR), which is a string of bits that represents a student's grade in that subject. The university keeps a database of all AGRs ever issued. A student can also obtain from his university an Authenticated Transcript/Diploma (ATD), which is a cryptographically signed version of his transcript or diploma. The US Department of Education (USDE) maintains a list of university public keys and issue dates as well as a list of revoked public keys.

Our design is based on industry standard design principles [7] with a strong emphasis on explicitness and open systems. AGRs are completely self-contained, allowing the student and the university to share responsibility for maintaining verifiable copies. All private keys (university and professor) are destroyed after the semester in which they are used.

2 Design Description

Private keys of universities and professors are used to generate certificates employed in AGR and ATD creation. To verify AGRs and ATDs, the public keys are obtained from the USDE. While retrieving university public keys, verifiers can also check whether the relevant public keys appear on the latest key revocation lists. However, in order to use any of the algorithms described below for AGR and ATD verification, a verifying entity (such as a graduate school) must make several assumptions concerning which parties to trust.

Users of the AGR system are expected to take reasonable precautions to safeguard private information, even though the AGR system does not mandate such precautions. For example, a professor could easily send a student an AGR using unencrypted email over an insecure network. Such a transmission would not affect the authenticity guarantees provided by the AGR system, but would compromise the student's privacy (and likely violate the Family Educational Rights and Privacy Act). Mechanisms used to provide additional guarantees beyond those set forth in the specification are beyond the scope of this work.

2.1 AGR and ATD Formats

As shown in Figure 1, the AGR consists of a header and a signed body. The header indicates the protocol version used to construct it as well as the professor's public key with which it was signed. The body consists of a Subject Certificate and Grade Information and is signed with the professor's private key. The Subject Certificate is a statement issued and signed by the university stating that a given student is taking a particular subject over a particular time period with a given professor. The Grade Information includes the same information as the Subject Certificate as well as the actual grade awarded in that subject. This redundant information ensures that if the professor mistakenly signs the wrong Subject Certificate, the AGR will be invalid.

Note that when the university refers to the professor, it always includes a copy of the corresponding public key in addition to the name. Since students do not have keys in this system, we use a Student Identifier that includes the student's name, gender, date of birth and Social Security Number or other identifier.

$$AGR = \left\{ \overbrace{\begin{matrix} ProtocolVersion \\ K_{Professor} \end{matrix}}^{Header}, \overbrace{\left\{ \begin{matrix} \overbrace{\left\{ \begin{matrix} ProtocolVersion, \\ StudentIdentifier, \\ ProfessorName, \\ K_{Professor}, \\ UniversityName, \\ DateofIssue, \\ SubjectDateRange, \\ Subject \end{matrix} \right\}_{K_{University}^{-1}}}^{SubjectCertificate}, \overbrace{\left\{ \begin{matrix} ProtocolVersion, \\ StudentIdentifier, \\ ProfessorName, \\ K_{Professor}, \\ UniversityName, \\ DateofIssue, \\ SubjectDateRange, \\ Subject, \\ Grade \end{matrix} \right\}}^{GradeInformation} \end{matrix} \right\}_{K_{Professor}^{-1}}}^{Body} \right\}$$

Figure 1: AGR Format

Figure 2 shows the format of the ATD. Like the AGR, it consists of a header and a body. The header includes the protocol version being used, the issuing university's name and the date of issuance. The ATD body is a statement signed with the university's private key attesting that a given student is in a particular degree program. It also includes a list of AGRs for all subjects taken by the student, as well as additional information such as a grade point average. Finally, the ATD includes a field indicating whether a degree has been completed. When this field is set, the ATD serves as a digital diploma; otherwise, the ATD is simply a digital transcript.

The AGR system makes use of the RSA public key algorithm for cryptographic operations. AGRs and ATDs are written as Extensible Markup Language (XML) using a standard Document Type Definition (DTD). Since XML makes use of Unicode, the AGR system can easily accommodate foreign language names and transcripts.

$$ATD = \left\{ \overbrace{\begin{matrix} ProtocolVersion \\ UniversityName \\ DateofIssue \end{matrix}}^{Header}, \overbrace{\left\{ \begin{matrix} ProtocolVersion \\ StudentIdentifier \\ UniversityName \\ DateofIssue \\ DegreeType \\ DegreeStatus \\ AdditionalInformation \\ ListOfAGRs \end{matrix} \right\}_{K_{University}^{-1}}}^{Body} \right\}$$

Figure 2: ATD Format

2.2 AGR and ATD Creation

AGR creation follows directly from the AGR format described above. The university issues one Subject Certificate to the subject's professor for each student in that subject. The university creates a Subject Certificate by assembling the required information (see Figure 1) and signing this information with the university's private key. Then, the university electronically sends the Subject Certificate to the professor.

Thus, in order to create an AGR, a professor must follow these steps:

1. Obtain a Subject Certificate from the university (see Figure 1)
2. Assemble the Grade Information
3. Sign the combination of the Subject Certificate and Grade Information with $K^{-1}_{Professor}$
4. Package the resulting document with a header indicating $K_{Professor}$
5. Send copies of the completed AGR to the student and the university

After obtaining a Subject Certificate, the professor assembles the Grade Information and signs the combination of the Grade Information with the Subject Certificate using $K^{-1}_{Professor}$. Finally, this signed information is packaged with the AGR header. After creating AGRs, professors must send copies to their university, since a university must store all the AGRs that each of its students (past and present) has received. These AGRs can be used on their own, or to build transcripts and diplomas, as we describe next.

In order to create an ATD, a university must follow these steps:

1. Assemble the required information in the ATD Certificate (see Figure 2)
2. Sign this information with $K^{-1}_{University}$
3. Package the combination with a header including the university name and the date of issue
4. Send the ATD to the student

After receiving an electronic request from a student, the university determines whether it can grant the request. If the university agrees to issue the ATD, it assembles an ATD Certificate as shown in Figure 2. Then, this information is signed with the university's private key and packaged with a header consisting of the protocol version, the university's name and the date of issue.

2.3 AGR and ATD Verification

To verify an AGR, a party should:

1. Verify the signed AGR body using the $K_{Professor}$ in the AGR's header
2. Obtain the public key of the university specified in the Grade Information from the USDE
3. Check with the USDE that the $K_{Professor}$ is not a revoked key
4. Check that $K_{Professor}$ in the header is the same as that in the Subject Certificate
5. Check that all common fields in the Subject Certificate and the Grade Information match

6. Check that the student presenting the AGR actually is the student described by the AGR's Student Identifier

After unsealing the body of the AGR with the $K_{Professor}$ from the header, the verifier should ensure that the header's $K_{Professor}$ matches the $K_{Professor}$ found in the AGR body's Grade Information. If they differ, the AGR is invalid. Next, the principal should obtain the public key of the university that signed the Subject Certificate by sending an electronic request to the USDE. The verifier should then check with the USDE that the signing key is not on a list of revoked keys. The request should include the university's name, the dates of issue and $K_{Professor}$, all of which are found in the AGR body's Grade Information. Upon receiving this request, the USDE looks up $K_{University}$ based on the university's name and the dates given. If the key cannot be found or is on the revoked list of university keys, then the USDE notifies the requester. Otherwise, the USDE sends the appropriate public key. The USDE also checks if $K_{Professor}$ is on the list of revoked keys and notifies the requester if so.

To verify an ATD, a graduate school or other entity should:

1. Obtain the public key of the university that issued the ATD from the USDE
2. Verify the ATD body signature using the university's public key
3. Check that the information in the ATD header matches the corresponding information in the ATD body
4. Verify all of the AGRs in the ATD as described above
5. Check that the student presenting the ATD actually is the student described by the ATD's Student Identifier

An entity obtains the public key of the university that issued the ATD by sending an electronic request to the USDE. The request and response protocol should be identical to the one used for obtaining $K_{University}$ for an AGR's Subject Certificate. After the verifier verifies the ATD Certificate supposedly signed with $K^{-1}_{University}$, they must verify that the signed and unsigned versions of the university's name and the dates of issue are the same. Furthermore, the entity should verify that the student who submitted the ATD is really the student described by the Student Identifier contained in the signed portion of the ATD.

2.4 Trust Considerations

The algorithms described above require communication between the USDE, universities, professors and students. However, any entity (such as a graduate school) verifying an AGR or ATD must make several assumptions concerning how sensitive material is kept secure and confidential during storage or transportation. Figure 3 presents a list of assumptions any verifying entity makes by trusting an AGR or ATD.

2.5 Key Generation, Distribution and Revocation

The AGR System requires the creation and distribution of different public-private key pairs every six months. At these times, all professors grading students will generate a new public-private key pair and distribute the public key to their university. Universities will also generate a new key pair and distribute their public key to the USDE. After all AGRs and ATDs have been issued

Assumption	Associated Principals
Student Information is being kept private	Student, Issuing University, Issuing Professor
$K^{-1}_{Professor}$ is secret	Issuing Professor
$K^{-1}_{University}$ is secret	Issuing University
Software and system security for AGR and ATD generating computers	Issuing Professor, Issuing University
Student Identifiers are accurate	Issuing University
Transport security for AGRs and ATDs	Issuing University
Transport security for $K_{University}$ keys and Key Revocation Lists	USDE and Verifying Party
Verification software and system security	Verifying Party

Figure 3: Assumptions needed to trust an AGR

for that six month period, the professors and universities will destroy their private keys to prevent compromising old data. If a previous term's grade needs changing, then new AGRs are issued using the current set of keys. Storage requirements for the different principals are listed in Figure 4.

If a professor discovers that his private key is compromised, he needs to immediately inform the university, who in turn reports all compromised professor public keys to the USDE. After creating a new key pair, the professor needs to re-issue the grades for that subject for that term. If the university private key is compromised, then the university needs to inform the USDE.

The USDE serves as a central authority that provides up-to-date information to individual schools. The USDE maintains an updated list of compromised university public keys and professor public keys as well as list of valid university public keys with corresponding dates.

While students don't have keys in this system, they do have identifying information. Since the security of the system depends on the accuracy of the Student Identifier data, universities should exercise great care in verifying students' identity upon admission.

Principal	Items to be stored
US Department of Education	List of ($K_{University}$, Date of Issue) pairs, List of revoked $K_{Professor}$ keys, List of revoked $K_{University}$ keys
University	$K^{-1}_{University}$, List of $K_{Professor}$ keys, AGRs for students
Professor	$K^{-1}_{Professor}$
Student	List of AGRs

Figure 4: Storage Responsibilities for all Principals

3 System Usage

3.1 System Goals

We now discuss how the AGR system presented in Section 2 meets system goals. Using Subject Certificates signed by his university, an instructor can produce AGRs on his computer. By consulting the USDE, graduate schools can verify the authenticity of AGRs. Graduate schools can also determine if a student has received a degree and if they are viewing all of the grades of a student. The system can also be used by students taking subjects from more than one undergraduate school,

by students taking subjects more than once and by universities changing a student's grade after issuing an AGR. Furthermore, unless one of the assumptions in Figure 3 fails, the AGR is valid for and can last as long as the student's entire life. Finally, the system is relatively easy to understand and use.

3.2 Instructor AGR Creation

As seen in Section 2.2, an instructor can create AGRs at the end of each semester using the software on his own computer, Subject Certificates obtained from his university and the $K^{-1}_{Professor}$ he generated. In this system, only Subject Certificates and $K^{-1}_{Professor}$ need to be available before AGRs are created on the instructor's computer. Another algorithm considered was for the professor to send all relevant information to his university (using software on his computer) and have that information checked and signed by $K^{-1}_{University}$, thus generating an AGR. However, we felt this alternative unduly centralized the creation of AGRs and thus failed to meet the specifications.

3.3 Verification

As seen in Section 2.3, a graduate school can verify the authenticity of an AGR by consulting the USDE. In this system, the graduate school communicates with the USDE to obtain the public key of the university that signed the Subject Certificate and check that $K_{Professor}$ is not a revoked key. The graduate school can decide if the AGRs that a student includes in an application really belong to that student by verifying the Student Identifier inside the AGR's Grade Certificate against the information presented by the student.

We also considered an alternative approach where graduate schools could get $K_{University}$ and check $K_{Professor}$ with the university that signed the Subject Certificate. However, we rejected this method because it requires that more trust be placed in the issuing university, as well as the communications channel between them and the verifier. In addition, this scheme requires that issuing universities exist in perpetuity, an unlikely event at best.

If a graduate school wants to check that an applicant has actually graduated or that an applicant has supplied grades for all the classes she has taken, the graduate school can verify the applicant's ATD. After verifying the ATD with the algorithm described in Section 2.3, graduate schools can simply examine the Degree field (see Figure 2) inside the ATD body to see if the student has a degree from the ATD-issuing university. Furthermore, the AGR list within the ATD can act as a complete, up-to-date record of all the AGRs that a student has received while attending the ATD-issuing university. After verifying these AGRs, graduate schools can be certain that they've seen every grade the student has received from the ATD-issuing university.

3.4 Corner Cases

The AGR system can handle many of the corner cases that arise in undergraduate education. For example, because a university stores all of the AGRs a student has received while at the university, the university can continually accept AGRs for subjects the student has taken more than once. There are no conflicts between these same-subject AGRs because the Subject Date Range field of each AGR's Grade Certificate and Grade Information would be different. When it issues an ATD, the university can include all of the AGRs it has for the student, including same-subject AGRs.

When a university wants to change a student's grade after issuing an AGR, the university should delete the old AGR from its database and replace it with a new AGR that includes the corrected grade. Since the university controls (through Grade Certificates) whether a professor can assign

a grade, the university also controls who signs the Subject Certificate and Grade Information combination. Thus, instead of $K_{Professor}$ and $K^{-1}_{Professor}$ being within the AGR, anyone with authority over issuing Subject Certificates could substitute their own K within the AGR and sign with the corresponding K^{-1}.

When a student takes subjects from more than one university, these universities separately issue AGRs to the student. Like same-subject AGRs, these AGRs will not conflict because the university name within each AGR will be different, as will the $K_{University}$ needed to verify each Subject Certificate.

3.5 AGR Lifecycle

As long as the assumptions in Figure 3 are valid, a student's AGRs and ATDs can be used throughout his lifetime. Because AGRs and ATDs can be verified by communicating only with the USDE, graduate schools and other ATD-receiving entities can authenticate ATDs and AGRs as long as the USDE still exists.

A student's AGRs and ATDs will probably last as long as his lifetime because both the student and the university share the responsibility of storing his grades. Since universities already store grades for these students, these universities can use the same back-up plans to store AGRs. For example, AGRs and ATDs can be stored at several separate, remote locations to decrease the likelihood of catastrophic data loss. At the same time, students can store their AGRs and ATDs in secure places like safety deposit boxes. Thus, in order for a student's AGRs and ATDs to disappear, every record that both the university and the student have would need to be destroyed.

3.6 Usability

Since the AGR system is based on public key encryption, implementors can draw on a wide body of knowledge accumulated over the last twenty years on how to build such systems. Furthermore, the system is flexible because participating universities control the transportation of sensitive materials and student and professor verification. This flexibility allows them to use their own computing systems to participate in the creation and verification of AGRs and ATDs. The verification of AGRs and ATDs is also aided by the inclusion of $K_{Professor}$ in AGRs, as this inclusion removes the burden of the USDE storing the public keys of professors. Finally, universities can minimize the transit cost of sending Grade Certificates through batching: At the end of a term, the university could automatically send Subject Certificates for each subject in which a student is registered. The hierarchy of the AGR system gives universities control over these optimizations, since they are responsible for the transportation of all sensitive material between themselves and their professors.

4 Failure Modes

In order to evaluate the security guarantees provided by the AGR system, we must consider both active attacks that deceive individual parties and denial of service attacks that inhibit use of the system. At the same time, security analysis must be conducted within the context a particular threat model. For the AGR system, that threat model must include computer science students who know how to break systems and have access to both professor and university computers. Students can be sophisticated adversaries with access to a wide array of penetration tools and the skill to create new ones. While they may lack the financial resources that large governments and corporations can muster, the prospect of graduate school admission provides them with significant

financial incentive to subvert the system. We discuss several vulnerabilities in the AGR system below.

4.1 Client Security

Because the AGR system is built upon public key encryption, it suffers from the same flaws that plague public key systems. These problems stem from the fact that people cannot remember cryptographic keys and cannot perform cryptographic computations mentally. Consequently, they must rely on their local computers to store keys and perform cryptographic operations. However, local storage of private keys makes them vulnerable to theft, replacement and deletion. Furthermore, delegating cryptographic operations to the computer means that anyone who can subvert the client computer can impersonate the principal on whose behalf the computer operates. As a result, the security of the entire system relies on the security of the individual client machines used by universities, professors, the USDE and anyone attempting to verify an AGR or ATD.

Stolen keys do varying levels of damage depending on the authority of their original owner. A stolen professor's key would allow an attacker to forge AGRs for students enrolled in whatever subjects the professor was teaching during one semester. On the other hand, a stolen university key would allow an attacker to forge transcripts and diplomas as well as arbitrary AGRs.

Given the dire consequences of key compromise, it is imperative that client computing environments remain secure. Unfortunately, evidence to date suggests that this is impossible, both because client computer systems cannot be secured and because the vast majority of computer users (even technical professionals) are unable to practice effective security habits [2]. For example, despite countless lectures on the need to select strong pass phrases, many users continue to use weak or easily guessed passwords [4]. The recent proliferation of email viruses and worms [11] illustrates just how easily client computers can be subverted [9], in other words, how easily the trusted computing base can be compromised [7].

4.2 Detecting Comprised Keys

This system has no mechanism to detect when a private key has been stolen and used to generate a forged AGR. While there are special cryptographic protocols that require the signer's participation in the verification process [8], we rejected these both because of their complexity and because AGRs could not be verified if the issuing university went out of business. We also considered compelling verifiers to pass copies of AGRs being verified back to the issuing university, where they could be compared with a list of all AGRs issued by the university. An AGR submitted for verification that was not on the issuing university's list of issued AGRs must have been forged with a compromised key. This system would allow the issuing university to detect compromised keys and update their key revocation lists accordingly.

While universities may agree to share verified transcripts with each other voluntarily, forcing verifiers to do so would greatly increase the complexity of the protocol as well as the demands placed on issuing schools. An important feature of the current system is that issuers only have to deal with a single central authority; forcing them to deal with each issuing university individually provides many more opportunities for attack. Finally, since schools that have gone out of business can not participate, it seems unwise to have universities rely on a security mechanism that may not exist in the long term.

In order to compensate for the system's inability to detect compromised keys, the USDE should encourage forensic computing initiatives so that computer system compromises can be detected without explicit support from the AGR system.

4.3 Key Distribution Vulnerabilities

The AGR system inherits another weakness from it's public key base: susceptibility to man in the middle (MITM) attacks. These attacks target the communications link between the USDE and a party trying to verify an AGR or ATD. An attacker impersonates the USDE and responds to requests for the issuing university's public key with her own public key. This allows an attacker to convince verifiers that an AGR created by the attacker purporting to be from a university actually belongs to that university.

Alternatively, the attacker can claim that a particular professor's public key has been compromised by appending it to the key revocation list. The verifying party would then conclude that the student providing the AGR had stolen the professor's private key and used it to create his own AGR, since innocent students would have new AGRs issued to them by the professor as soon as he realized his keys were compromised. The resulting confusion would be difficult to resolve, since there is no way for the student to convince the issuing university that anything is wrong.

MITM attacks are not limited to cryptographic protocols; they can be used against postal mail service and telephone conversations. This means that we cannot resolve these problems by making public keys available over the telephone. As we describe in Section 4.4, "secure" web sites on the public Internet are just as vulnerable.

Unfortunately, MITM attacks are difficult to defend against; the underlying problem, ensuring that clients have access to the server's public key, has proven to be the bane of public key protocols [8]. Nevertheless, there are techniques verifiers can use to reduce the likelihood of a MITM attack. One such technique is to store university public keys provided by USDE indefinitely. If the verifier finds the USDE has silently changed the public key of a university, it can assume that it has not been communicating with only the USDE. This is the same approach used by the Secure Shell protocol [13].

Another technique is to corroborate public key lists from several sources. Mismatches between sources indicate that at least one source is an attacker impersonating the USDE. Attackers must now impersonate the USDE in several different contexts which can be much more difficult. For example, the USDE could ship CD-ROMs of public key lists to all universities. These CD's would include a cryptographic hash of their contents written on the front cover. Professors could verify that the hash displayed on the label corresponds to the disk contents by calculating it themselves. They could then compare CD hashes with distant colleagues during conferences. Unless an attacker could intercept and replace the original disks of all professors, the group would be able to detect the discrepancy and conclude that at least one of them did not have the correct set of public keys.

4.4 Failure of the Public Key Infrastructure

Although not directly implicating the AGR system, the complete failure of large scale PKI systems bodes poorly for it. Despite years of effort and significant investment, the largest and most well known PKI, that used by the public Internet to certify web sites for HTTP [3] over TLS [5], has proven vulnerable to a large number of attacks[6]. These include MITM attacks using DNS poisoning or ARP spoofing as well as simpler semantic attacks (offering links to a slightly misspelled version of a trusted site) and social engineering attacks. In fact, simple tools exist to facilitate MITM attacks [12]. The problems inherent in the underlying public key protocols are exacerbated by the complete failure of vendors to implement security-critical components of the public key infrastructure. For example, the vast majority of web certificates have no real CRL mechanism in place [10]. We decided to base the AGR system on public key cryptography, because, despite its many flaws, it is the best available technology for the job.

4.5 Scalability

As was originally designed, the AGR system did not scale well as the number of students in a particular subject grows. While many subjects are relatively small, a significant minority may have several hundred students enrolled at a time. The professors in such subjects will have to issue several hundred AGRs, a process that is tedious, time consuming and error prone.

We considered using hierarchy to solve this problem in much the same way universities use hierarchy to solve teaching scalability problems. Professors teaching large subjects rarely work alone; instead, they rely on a layer of recitation instructors or teaching assistants. The end result is that faculty members at the lowest level of the hierarchy only deal with a manageable number of students. We can implement this notion of hierarchy by generalizing the notion of delegation used in AGRs. In particular, the university will now provide separate signed statements indicating that a student is enrolled in a particular subject and that a particular professor has authority to issue grades for a certain subject. Professors in large subjects can make signed statements claiming that particular recitation instructors have authority to issue grades for a given subject in a given semester, perhaps for a particular subset of students.

The beauty of this system is that the university does not need to know how the professor delegates his authority; each layer only needs to communicate with the layer immediately above and below it. A second benefit is that it allows for finer grained control since the most powerful keys (the lecturer's) are only used infrequently to delegate authority to recitation instructors. The keys used most often are most limited. While conceptually elegant, we rejected this authority delegation scheme because of the extra complexity it imposes. It requires more keys to be used and makes it harder to reason about the protocol and audit particular AGRs. It appears that an implementation to verify AGRs would be sufficiently complicated to introduce many new bugs.

The alternative we settled on was to have universities specify recitation numbers in addition to course numbers in all certificates for large subjects. Thus, the course field in the Subject Certificates and Grade Information always includes a recitation number. This solution allows individual recitation instructors to grade students in their sections without giving them the ability to issue AGRs for the entire subject, thus providing finer grained control of keys.

4.6 Single Point of Failure

Since the USDE acts as a trusted authority, maintaining university public keys and revoked key lists, it is a single point of failure. As such, denial of service attacks targeting the USDE would severely compromise the AGR system. In addition, centralized databases that concentrate so much power provide very lucrative targets to attackers since the potential reward is so high. Thus, the USDE suffers from many of the same problems that plague central key authorities in key-escrow proposals [1].

As an alternative, we considered using a web of trust model where any party could store public keys and revocation lists. Verifying parties would exchange public keys in person with their nearest neighbors. By exploiting the transitive nature of trust relationships, a verifying party could discover the public key of any university in the system. We rejected this approach because this degree of decentralization makes it impossible for the government to make guarantees about the security of the system.

5 Conclusions

The AGR system provides the necessary infrastructure for paperless grade reports, transcripts and diplomas. Its design is based on key principles of simplicity, explicitness and openness. Because of the emphasis on openness, the only secrets present are secret keys, which have a limited lifetime and can be revoked. All other information can be published without compromising the security of the system. As a result, the AGR system requires that communications only be authenticated, not necessarily confidential. The focus on explicitness is visible in the repeated fields shared by Subject Certificates and Grade Information sections, as well as ATDs and AGRs. The repeated use of protocol version and date of issue fields help ensure that any particular message is both fresh and appropriate [7] for the context in which it was received.

We strove for simplicity so as to make the AGR system easy to understand, verify, implement and audit because doing so offers the best hope for security in the long term. As part of the drive to simplify, we endeavored to control complexity by using hierarchy to minimize the interactions between different entities. In the AGR system, information only passes between an entity and nodes immediately above and below it in the hierarchy. Professors only deal with their parent university, not the USDE or other schools, while universities deal only with their own professors and the USDE. Verifying parties deal with the USDE alone, not with individual universities or professors.

Secondary design principals include the end-to-end argument, threat model-based analysis and an acute recognition of our own limits. Support for the end to end argument derives from users' ability to verify AGRs and ATDs with as much confidence as they desire: greater confidence requires greater effort, but users are not forced to accept uniform authenticity guarantees. In other words, we wholeheartedly repudiate "one size fits" all approaches to security. Inevitably, such uniform guarantees are watered down to the lowest common denominator leaving sites requiring strong guarantees unsatisfied or made so restrictive that the majority of careless sites are unwilling to adopt the standard.

Based on the threat model and our own limits, we premised the design on the notion that security failures will occur in the real world. Effort should be directed at mitigating the effects of those failures rather than on futilely trying to completely eliminate them. For similar reasons, we chose to draw on the twenty years of community experience designing, building and deploying public key cryptography systems rather than creating entirely new protocols or ciphers.

References

[1] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, and Bruce Schneier. The Risks of Key Recovery, Key Escrow, and Trusted Third Party Encryption. http://www.cdt.org/crypto/risks98/, 1998.

[2] Ross Anderson. *Security Engineering*. Wiley Computer Publishing. John Wiley & Sons, 2001.

[3] T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol — HTTP/1.0, May 1996. Status: INFORMATIONAL.

[4] Andrew Brown. UK Study: Passwords often easy to crack. http://www.cnn.com/2002/TECH/ptech/03/13/dangerous.passwords/index.html\%, March 2002.

[5] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, January 1999. Status: PROPOSED STANDARD.

[6] Carl Ellison and Bruce Schneier. Ten Risks of PKI. *Computer Security Journal*, 16(1):1–7, 2000. Also available at http://www.counterpane.com/pki-risks.html.

[7] Jerome H. Saltzer and M. Frans Kaashoek. *Topics in the Engineering of Computer Systems*. Massachusetts Institute of Technology, Cambridge, MA, 2002.

[8] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[9] Bruce Schneier. Why Computers are Insecure. *Cryptogram*, Novermber 1999. Also available at http://www.counterpane.com/crypto-gram-9911.html\#WhyComputersareInsecur\%e.

[10] Roy Sinclair. Verisign Certificates Problem. *Forum on Risks to the Public in Computers and Related Systems*, 21(30), March 2001. Also available at http://catless.ncl.ac.uk/Risks/21.30.html\#subj7.

[11] Robert M Slade. More on klez. *Forum on Risks to the Public in Computers and Related Systems*, 22(5), May 2002. Also available at http://catless.ncl.ac.uk/Risks/22.06.html\#subj6.

[12] Dug Song. dsniff Frequently Asked Questions. http://www.monkey.org/~dugsong/dsniff/faq.html\#HowdoIsniff/hijackHTTPS/SSHconnections, December 2001.

[13] Tatu Ylonen. SSH — Secure Login Connections Over the Internet. In *6th USENIX Security Symposium*, pages 37–42, San Jose, CA, July 1996.