Oblivious Key Escrow

Matt Blaze
AT&T Research
Murray Hill, NJ 07974
mab@research.att.com

12 June 1996

Abstract

We propose a simple scheme, based on secret-sharing over large-scale networks, for assuring recoverability of sensitive archived data (*e.g.,* cryptographic keys). In our model anyone can request a copy of the archived data but it is very difficult to keep the existence of a request secret or to subvert the access policy of the data "owner." We sketch an architecture for a distributed key escrow system that might be suitable for deployment over very large-scale networks such as the Internet. We also introduce a new cryptographic primitive, *oblivious multicast,* that can serve as the basis for such a system.

1 Introduction

In any system in which sensitive information must be stored for future use there is a fundamental tension between ensuring the *secrecy* of data against those who are not authorized for access to it and ensuring its continued *availability* to those who are. Secrecy is often best served by making only a small number of carefully-guarded copies of the data, while availability favors a policy of the widest possible dissemination in the hope that at least one copy will be intact at the time it is required. In general, a balance has to be struck between these two goals based on the requirements of and resources available to the particular application, but in any case copies of the sensitive data must be controlled in some careful manner (*e.g.,* through the use of an off-site, trusted backup facility employing guards and other effective, if expensive, security practices).

Another approach is "key escrow," in which sensitive data are encrypted so that the ciphertext can be widely copied and backed-up via conventional methods, but the decryption keys are controlled in some careful manner by trusted third parties who assume responsibility for revealing the keys to authorized entities in the event of an emergency. One advantage of escrowing keys instead of the raw data is the flexibility to perform the escrow at any time, even prior to the actual creation of the data, and the ability for one escrowed key to represent arbitrarily much encrypted information. A number of key escrow schemes have been proposed for a variety of applications, most with the aim of facilitating law enforcement access to encrypted data, but also for commercial data recovery [NIST94, WLEB96, Denn96].

Third party backup, whether of data or keys, has a number of disadvantages, however. The "escrow agents" must be highly trusted and carefully protected, since compromise of a single escrow site (or small set of sites, in the case of split data) can result in an irrevocable loss of security. Since protecting such data is likely to be expensive, one escrow site can be expected to serve many different sets of data, making each site an attractive "fat target" for attack. Finally, legal, liability, and conflict-of-interest issues sometimes make it difficult to ensure that an escrow agent will act only in the best interests of the data owner, especially when served with a legal demand to turn over keys or tempted with some inducement to misbehave. One of the frequently-raised objections to government-run key escrow systems (*e.g.,* the "Clipper" chip) is the fear that the escrow centers will, perhaps secretly, assist a rogue government in violating its citizens' privacy.

In this abstract, we propose a different model for assuring both recoverability and protection of sensitive data based on two concepts: secret-sharing and the decentralized nature of large, heterogeneous networks such as the Internet. In our model, anyone can request a copy of anyone's data but it is not possible to keep the existence of such a request secret or to subvert the access policy of the data "owner" without subverting a significant fraction of participants in the network. There are no explicit "escrow agents"; instead, key shares are distributed widely to ordinary networked computers spread across a wide variety of administrative and geographic boundaries.

2 "The Net" as an Escrow Agent

The goal of our scheme is to make it difficult to recover escrowed data without the knowledge and consent of its owner, while still assuring high availability in an emergency. Its security rests on the premise that highly distributed systems spread over many administrative, political, and geographic domains (such as the Internet), are more robust than any single site or small set of sites, no matter how well protected. Other systems, such as Eternity [Ande96], have recognized and exploited this property of global networks for maintaining information availability; we simply expand this notion to include secrecy as well.

We assume that each node (or a large fraction of nodes) in the network runs an "escrow server" that performs most of the steps of the protocol, and that there is some broadcast mechanism for reaching them (which could be based on existing mechanisms such as Usenet news). The first step in using "the net" as an escrow agent is to split the key to be escrowed using some secret-sharing scheme [Simm92] with a very large number of shares (*e.g.,* a k-out-of-n threshold scheme where $k = 500$ and $n = 5000$, but we leave the details of determining an appropriate access structure to the reader). Next, we package each share along with a key identifier, a digital signature of the share, and a policy describing the circumstances under which the share should be disclosed (discussed below). Finally, we select, at random (or according to some other policy) as many sites as we have shares and send one share to each site, over a secure channel. We then destroy the shares and the list of sites to which they were sent.

To recover escrowed data, we broadcast a request for shares for the key identifier we want to recover, using some mechanism that is likely to be received by the shareholders' escrow servers. Upon receipt of a request for shares, each escrow server logs the request and, if it holds a share for the key in question, checks the policy contained in the share package. If the request conforms to the policy we send the share to the requester. The requester (who can verify the authenticity of each share by checking the signature) can recover the key once enough shares have been received.

Whether such a scheme is robust, secure, or otherwise adequate depends primarily on three factors: the reliability (with respect to continued availability, security against compromise, and ability to follow instructions) of the nodes that handle the key shares, the access structure of the secret-sharing scheme, and the nature of the policy that each node is supposed to follow.

If the nature of today's Internet is any indication, we must assume that the individual nodes are not very reliable, especially over time. Some nodes will simply disappear. Others will maliciously fail to follow instructions. Still others will fail to safeguard their shares, sometimes due to malice but more often as a result of mistake, incompetence, or failure of some underlying security mechanism. It is likely that as the net grows these issues will become even more pronounced. Therefore, the security of the scheme depends on a choice of access structures and policies that assumes that a large fraction of shareholders will not follow the correct protocol.

The secret-sharing access structure must be chosen to require enough shares to prevent key recovery by collusion among a few nodes, yet with enough redundancy to allow recovery in the likely event that most nodes are not available or did not retain their shares at the time key recovery is required. Unlike almost all other problems in distributed computing, scale appears to actually help here; consider, for example, a 500-out-of-5000 threshold scheme, which permits key recovery even when 90% of nodes have failed and yet retains its security until at least 500 nodes have been compromised. The distribution of nodes could also play a part here, particularly when the key is split with a more sophisticated access structure. For example, key shares could be distributed to nodes selected across a variety of administrative, legal, political and geographic domains, with the access structure selected to require that shares be collected from nodes in several different categories.

Each shareholder is asked to enforce the access policy included with the share. The policy must be designed to facilitate emergency access without also permitting undetected disclosure. Because shares can only be recovered by broadcasting, we can take advantage of the inherently public nature of requests in formulating the access policies. For example, the policy might specify a public signature key (to which the real key owner knows the corresponding secret) and instructions to delay revealing key shares for some period of time, say one week. If an unauthorized request for a key is broadcast, the legitimate key owner would have one week to notice the request and broadcast another message, signed with the signature key, requesting that the shareholders ignore the original request and turn over information that might aid in tracking down the source of the unauthorized request. Policies might also include instructions on the minimum identification that share requests must include and instructions on how share requests should be logged (*e.g.,* by posting to a news group or even advertisements in newspapers). They might also include an expiration date beyond which the share is to be deleted. We defer the question of how policies should be specified, but it may be sufficient for the server, upon receipt of a share request, to send a message to its (human) operator containing instructions (written in English) that were included in the share package.

It may also be desirable to obscure the nature of the data held by each shareholder from the shareholders themselves. Key identifiers might best be chosen so that an outside attacker cannot derive the purpose or owner of the key from its identifier and so that shareholders do not know exactly what their shares are for. (One approach is to use a randomly generated key ID, long enough to be collision free, that is stored with the ciphertext. A more sophisticated approach involves using multiple key IDs for each key, generated from a random seed stored with the ciphertext, so that shareholders cannot determine whether they hold shares for the same keys as one another. Still another approach is to base the key ID on the signature key used to sign the shares.)

Some infrastructure is required. Key owners would need a directory or other mechanism for identifying and communicating with escrow servers at the time the shares are created. A broadcast mechanism for key recovery is also required. It is possible that existing mechanisms could suffice for both these purposes (*e.g.,* DNS for server identification and Usenet news for broadcasting) but more specialized systems would be required if this scheme were to be fielded on a large scale. Finally, of course, the escrow servers would need to be deployed widely, perhaps included as a standard feature in networked operating systems.

Share distribution must be secure against both eavesdropping and traffic analysis. The need for security against eavesdropping is obvious, since observing all the shares allows recovery of keys without the assistance of the shareholders. Resistance to traffic analysis is required to ensure that shares can only be recovered by broadcasting. If the identities of the shareholders are known, an attacker could "target" the sites believed to be weakest, and, if successful, recover shares without broadcasting the request and without following the share access policy. Ideally, shares would be distributed via a completely anonymous communication protocol in which neither sender nor receiver learn one another's identity, nor can an eavesdropper. Share distribution should be deniable as well; it should be infeasible for a third party to determine that a given node has sent or received a key share.

More formally, shares should be distributed via a cryptographic primitive we call *oblivious multicast.* In a k-out-of-n oblivious multicast, the sender sends a message to a list of n potential receivers from which he is guaranteed that at most k will actually receive it. Each potential receiver should not be able to "cheat" the protocol to increase its probability of receiving a given message to beyond k/n, and the sender's influence should be limited to selecting the list of potential receivers. Under this model, if the key is split into 5000 shares to be distributed throughout a network of 10^6 nodes, each share would be sent using a 1-out-of-10^6 oblivious multicast. We give a simple oblivious multicast protocol in the Appendix.

Even in the absence of a true oblivious multicast protocol, it may be sufficient in some applications simply for the sender to select, at random, each shareholder from among the nodes on the network and send each share via an anonymous communication scheme such as a "Mix" [Chau81]. Of course, the list of shareholders or potential shareholders need not, and indeed should not, be retained by the key owner once the shares have been distributed.

3 Emergency Access – "Angry Mob Cryptanalysis"

Under ordinary circumstances when key recovery is required, the original key owner will initiate the request. The owner extracts the key ID and broadcasts the request to the network, performing whatever (presumably public) logging is required by the policy that was sent to the shareholders. Upon receipt of the broadcast, each server checks whether it is a shareholder for the requested key. If it is, it checks whether request satisfies the access policy (perhaps by transmitting a copy of the English-specified policy to the server operator, perhaps by automated means if the policy is more formally specified). If the access policy is satisfied (*e.g.,* a message announcing the request appeared in some established place, a certificate of the identity of the requester was included in the request, or whatever) and after expiration of a policy-specified waiting period to allow for repudiation of the request by the legitimate key owner, the share is transmitted back to the requester over a secure channel. The requester can then combine these shares to recover the key; corrupted shares will not affect the protocol since legitimate shares were digitally signed by the original key owner at the time they were distributed.

Sometimes, however, an extreme emergency might make it necessary to recover keys in a manner contrary to the policy specified in the original shares. For example, it may be necessary to recover keys before the policy-imposed delay has elapsed, or to obtain access in spite of the objections of the original key owner. Such a situation is most likely to arise from some kind of law enforcement or public safety emergency in which the requester makes the case that public policy should supersede the access policy of the key owner. Of course, such a situation is fraught with difficult issues of judgement and policy, and fears of abuse, fraud, or coercion are among the primary objections raised against key escrow in general. Our scheme places the burden of determining whether exceptional access requests should be granted on the shareholders.

Indeed, the dependence on the collective judgement of the widely distributed shareholder operators may be the scheme's most important property. Under normal circumstances, the shareholders can be expected to behave approximately as specified in the share policies (with occasional pathological exceptions, limited in their effect by the nature of the secret-sharing access structure). In exceptional situations, however, a public appeal can be made in an attempt to convince the shareholders to reveal their shares in a manner not permitted by the stated policy (*e.g.,* the police could broadcast an appeal for key shares on television news, stating the facts of the case under investigation). In particular, because the identities of the shareholders are not known, such an appeal must be done publicly and in a manner designed to attract considerable attention. It is not possible to secretly induce, through legal means or otherwise, shareholders to reveal their shares. For some applications (*e.g.,* personal information associated with an individual), such a scheme could be acceptable even when key escrow is not. (We propose the rather lighthearted phrase "angry mob cryptanalysis" to refer to the threat of enough shareholders being convinced to violate the share access policy to permit key recovery. It is distinguished from "rubber hose cryptanalysis," in which keys are derived by means of legal or extra-legal coercion[1].)

4 Conclusions

Key escrow is a confusing subject, especially so because there is little general agreement as to even its basic goals and requirements. We have proposed a scheme that has a number of interesting properties that may make it appropriate for protecting secrecy and availability in certain kinds of applications. A number of open problems remain, of course, before such a scheme could be made completely practical. Areas for further study include the effects of different access structures, specification of policy, economic, performance and reliability analysis, and efficient protocols for large-scale oblivious multicast. The most challenging problems arise from the practical difficulty of introducing a standardized escrow service and deploying it on a large scale.

Of course, we do not in complete seriousness propose this scheme as a general solution to the key recovery problem, but hope primarily to open a new avenue of discussion. Although not designed specifically for law enforcement, the scheme appears to address many of the concerns voiced by critics of "government" key escrow as well as many of the (stated) concerns of law enforcement.

[1]The phrase "rubber hose cryptanalysis" appears to be due to Phil Karn.

5 Acknowledgements

Much of the inspiration for this scheme arose from Ross Anderson's description of the motivation and principles behind the Eternity service, in conversations at Cambridge and AT&T Bell Labs. We also thank Joan Feigenbaum, Yvo Desmedt and Raph Levien for their helpful comments, and the Issac Newton Institute for Mathematical Sciences, Cambridge, for hosting the author while most of this work was done.

References

[Ande96] Ross Anderson. "The Eternity Service." Invited paper to appear at *Pragocrypt 96.* 30 September – 3 October 1996, Prague.

[Chau81] David Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." *CACM.* February 1981.

[Chau82] David Chaum. "Blind Signatures for Untraceable Payments." *Proc. CRYPTO82.* August 1982.

[Denn96] Dorothy Denning. "A Taxonomy for Key Escrow Encryption Systems." *CACM.* March 1996.

[NIST94] National Institute for Standards and Technology. Escrowed Encryption Standard, *FIPS 185.* U.S. Dept. of Commerce, 1994.

[Rabi81] M. Rabin. "How to Exchange Secrets by Oblivious Transfer." *TR-81.* Harvard Aiken Computation Laboratory, 1981.

[Simm92] G.J. Simmons. "An Introduction to Shared Secret and/or Shared Control Schemes and their Applications." In *Contemporary Cryptology,* Simmons, ed. IEEE, 1992.

[WLEB96] Stephen T. Walker, Stephen B. Lipner, Carl M. Ellison, and David M. Balenson. "Commercial Key Recovery." *CACM.* March 1996.

Appendix: Oblivious Multicast

It is possible to build an approximation of k-out-of-n oblivious multicast out of conventional oblivious transfer primitives [Rabi81], *e.g.,* by performing, with each of n potential recipients, an oblivious transfer in which the message has probability k/n of being delivered correctly. The overall outcome of such a protocol is only probabilistically specified, however, since the outcome of each transfer is determined independently from the others.

We give here a non-probabilistic k-out-of-n oblivious multicast protocol, based on blind signatures [Chau82], in which the sender controls exactly the number of successful transfers but cannot learn who the successful recipients are from among the set of potential receivers. We illustrate the scheme with RSA, but it can be adapted trivially to any other blind signature scheme. We assume that there is an authenticated, secret channel between each pair of nodes in the network (*e.g.,* each node publishes trusted public signature and encryption keys), an anonymous communication mechanism that hides the identity of a message's sender, and a broadcast mechanism that allows one-way communication between one node and all other nodes.

The players include a sender, S, and a set of n potential receivers R. S generates (either once or, optionally, once for each multicast, if S wishes her identity to remain secret) an RSA key (S_{pub}, S_{priv}, m), where m is the modulus.

To begin the protocol, S publishes its public key and broadcasts to the members of R a request that they begin the protocol. Each node $R_i \in R$ selects, at random, a *key token* t_i and a *blinding factor* b_i (with inverse b_i^{-1}). R_i calculates and sends the blinded key token

$$\overline{t_i} = t_i(b_i^{S_{pub}}) \bmod m$$

to S, keeping both b_i and t_i secret. (R_i signs this message to establish its origin to S.) For each such message received, S first verifies that it has not previously received a message from R_i, calculates the blind signature

$$\overline{\alpha_i} = \overline{t_i}^{S_{priv}} \bmod m$$

and returns $\overline{\alpha_i}$ to R_i.

Upon receipt of $\overline{\alpha_i}$, R_i can compute the unblind signature of t_i by calculating:

$$\alpha_i = b_i^{-1}\overline{\alpha_i} \bmod m$$

When S has transmitted $\overline{\alpha}$ values to all members of R, each $R_i \in R$ sends the signed key tuple (t_i, α_i) to S, encrypted with S's public encryption key and using a communication mechanism that hides R_i's identity.

Note that once each member of R has completed this phase of the protocol, S has received n unique (t, α) signed key tuples (one from each $R_i \in R$). S can verify that each message came from a different member of R (by checking the uniqueness of the message and by verifying the unblinded signature α) but cannot determine the mapping of key tuples to individual nodes (because the signature was blinded). Thus each t serves as a secret key known only to S and a single, unknown, member of R.

Finally, to oblivious multicast to k members of R, S encrypts k copies of the message (using a symmetric-key cryptosystem), with a different t key selected at random from among the valid received key tuples for each. Each of these encrypted messages is broadcast to all members of R. The successful recipients are those who generated (and therefore know) the keys that S selected. An implementation can allow nodes to determine whether they ware successful in several ways. The simplest is to require each member of R to attempt to decrypt every message with its t. If the message follows a pre-determined format (*e.g.,* it includes a fixed value field of sufficient length that it is unlikely to have the correct value at random), the decrypted message can be compared against the expected form. Another, perhaps more efficient, option is to prefix to each message a one-way hash of the t with which it was encrypted.

The most obvious application of this protocol to key share distribution uses a separate 1-out-of-n oblivious multicast to distribute each share. Since each multicast requires two exchanges with every node, if there are m shares and n potential shareholders in the network a complete share distribution requires at least $2nm$ exchanges over the network. It is possible to eliminate the factor of m, however. Observe that it is not necessary in the last stage of the protocol that the same message be encrypted with each t. All shares could be distributed with a single pass of the protocol, with a different share distributed with each selected t. This optimization, which reduces communication to only $O(n)$ message exchanges plus $O(m)$ broadcasts, has the added virtue of making it possible to guarantee that each node receives at most one share.

Clearly, to be of practical utility for share distribution, the protocol must be efficient even when n is very large. The blind signature-based protocol above is of dubious efficacy if n represents, *e.g.,* every node on the Internet, but it could, depending on the processing and network cost model, provide adequate performance when used with a subset of nodes large enough to make it difficult to target the entire set of potential receivers.

We suspect that there are applications besides key share distribution that could make efficient use of an oblivious multicast primitive (*e.g.,* distribution of papers to reviewers, blind surveys, etc.).