RZ 2970 11/03/97
Computer Science/Mathematics 5 pages

Research Report

How to Break Fraud-Detectable Key Recovery

Birgit Pfitzmann

Universität des Saarlandes
Fachbereich Informatik
Postfach 151150
D-66041 Saarbrücken, Germany
email: pfitzmann@cs.uni-sb.de

Michael Waidner

IBM Research Division
Zurich Research Laboratory
Säumerstrasse 4
CH-8803 Rüschlikon, Switzerland
email: wmi@zurich.ibm.com

How to Break Fraud-Detectable Key Recovery[*]

Birgit Pfitzmann[α], Michael Waidner[β]

October 29th, 1997

Abstract. Fraud detection for software key recovery schemes means that, without knowing the session key, a third party can verify whether the correct session key could be recovered. This concept and a construction by so-called binding data was introduced by Verheul et al. at Eurocrypt '97 to provide for dishonest users that make simple modifications to messages, e.g., delete the key recovery information, and manipulate the recipient's software such that it decrypts messages even if the key recovery information is incorrect.

We show how to break their general construction within their model, in particular without using any other encryption system or any pre-established shared secrets.

We conclude that the concept of binding data does not improve the security of software key recovery but illustrates once more its fundamental problem: it does not improve an authorized third party's ability to eavesdrop on serious criminals.

1 Key Recovery

Consider the standard scenario of secret communication by means of encryption: A ciphertext *c* is created by Alice and sent to Bob. The key management system has to ensure that Bob, and only Bob, knows the right secret key *sk* to decrypt *c*.

Key recovery schemes voluntarily alter this scenario by introducing a third party, Tracy, who can recover *sk* from *c*, or at least a part of *sk* that is large enough so that the other part can be computed with moderate effort. For instance, Tracy might get the first 16 bits of a secret DES key and has to determine the other 40 bits by brute force. Other schemes require Alice or Bob to escrow secret information at Tracy in advance or during a session key agreement phase, or Tracy computes this key herself and distributes it to Alice and Bob. Several variants aim at improving the trust in Tracy: systems might offer Alice and Bob a certain choice of third parties; Tracy might be implemented in a distributed way, involving secret sharing techniques; and the functions of computing *sk* and actually decrypting *c* might be implemented by separate entities.

There are several intended applications as well as several risks for key recovery:

- ***Personal key recovery:*** Alice and Bob are actually the same person, the message is a file stored in Alice's encrypted file system, and she has forgotten *sk*. (If it were a real transfer and Bob had forgotten *sk*, Alice could usually repeat the message.)

  We consider this the practically most relevant application. However, solutions for this scenario could be much simpler than those primarily motivated by law enforcement, and one could leave the implementation and choice of Tracy entirely to Alice.

[*] Presented at the Rump Session of Eurocrypt '97, Konstanz, Germany, May 13, 1997.
The same attack was independently discovered by H. Tiersma, who gave a presentation "Unbinding ElGamal - An Alternative to Key-escrow?" at the same Rump Session.

[α] Universität des Saarlandes, Fachbereich Informatik, Postfach 151150, D-66041 Saarbrücken, Germany; <pfitzmann@cs.uni-sb.de>.

[β] IBM Research Division, Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland; <wmi@zurich.ibm.com>.

- ***Commercial key recovery:*** Bob's employer wants to read his data, e.g., because he has left the company or has suddenly fallen seriously ill. (But as long as Bob is trusted he could choose with whom to leave the key.)
- ***Law enforcement:*** Criminal investigation or national security agencies suspect Bob or Alice of having committed a certain crime and wish to eavesdrop on their messages.

For the law enforcement application one has to assume that Alice and Bob cannot cheat Tracy. Cheating means that they use another, more eavesdrop-resistant encryption technology, like PGP [Zimm_95] (at least Version 5.0 or earlier), or that they manipulate the key recovery scheme such that Tracy cannot recover the secret key. Doing this is easy to conceal by the well-known *superencryption*: After encryption with the recovery-resistant scheme, the ciphertext is encrypted again, using the "legal" scheme. Thus, the result looks "legal" but cannot be decrypted by Tracy.

The most critical security problem of key recovery is its potential for misuse (see also [AABB_97]): A criminal organization wanting to eavesdrop on Alice and Bob may achieve this by getting hold of Tracy's information. There are many ways to do this: forging court orders, misleading Tracy about the identities of Alice or Bob (i.e., Tracy might decrypt messages for someone else without noticing it [FrYu_95]), hacking into Tracy's computer system, stealing her backups. More likely is simple bribery of Tracy's employees or equipment providers, and a very dangerous situation is subversion of the political system. Whoever wants to deploy such schemes should carefully consider whether the expected benefit is worth the increased risk. This is particularly true for national key recovery schemes, which would offer strong criminals a single point of attack on the whole society.

To allow consistent cryptographic work on key recovery in spite of the superencryption attack, the typical model is that Alice and Bob have no previously exchanged secrets, cannot modify the software of the key recovery scheme, and that they have no other encryption system available [Desm_95, KnPe_96]. Even though in particular the third assumption is quite unrealistic, it would not count "breaking" a key recovery scheme to deviate from these assumptions. However, the model of key recovery with binding data is slightly different from the normal model, and we will break those schemes entirely within their own model.

2 Key Recovery with Binding Data

Most of the key recovery schemes recently proposed are based on hybrid encryption: A message m is not directly encrypted with Bob's public key, *pk_Bob,* but first a session key S is chosen, and then m is encrypted as $< E_S(m) \| E_{pk\_Bob}(S) >$, where $\|$ denotes concatenation. Now the key recovery agent Tracy is treated like an additional recipient, i.e., the ciphertext becomes $< E_S(m) \| E_{pk\_Bob}(S) \| E_{pk\_Tracy}(S) >$). Obviously, Tracy can decrypt m just like Bob.

2.1 Key Recovery with Binding Data

In [VeTi_97], a proof, *bind*, is added to the ciphertext that convinces any fourth party that the second and third component are encryptions of the same session key, S. The fourth party should not need any secret key and should not obtain any additional information about S.[1] Thus, c is now of the form

[1] The main part of [VeTi_97] shows in detail how to construct such a proof if ElGamal encryption is used for $E_{pk\_Bob}(S)$ and $E_{pk\_Tracy}(S)$. As our attack already works on the abstract level, we do not repeat this here.

$$c = < E_S(m) \,||\, E_{pk\_Bob}(S) \,||\, E_{pk\_Tracy}(S) \,||\, bind >.$$

The main purpose of this addition is to enable external observers, in particular the network operators, to filter incorrectly constructed ciphertexts, i.e., this is meant as a mechanism to enforce the use of the key recovery scheme.

2.2 The Model

In [VeTi_97 p. 121] an attacker model is described, and we use exactly this model for our attack. As in normal key recovery schemes, it is assumed that a priori Alice and Bob do not share any other encryption mechanism or secret key. In contrast to normal key recovery schemes, however, it is assumed that the key recovery scheme is implemented in software only and that Alice and Bob can make simple modifications to this software, in particular: [2]

- Alice can delete or destroy the key recovery components $E_{pk\_Tracy}(S)$ and *bind* from the ciphertext *c*.
- Bob can manipulate his decryption software such that the results of consistency tests are ignored. This means that if Bob enters $< E_S(m) \,||\, E_{pk\_Bob}(S) \,||\, x >$ with arbitrary *x*, the manipulated decryption software will output *m*.

3 Breaking Key Recovery with Binding Data

We first describe the attack. Afterwards, we discuss whether the schemes could be repaired, or rather, whether there would be another model more like that of original key recovery in which our attack would not count. However, any such model makes binding unnecessary.

3.1 The Attack

The attack is similar to the general superencryption attack that was excluded from the model. The variation is that, due to the assumptions the authors made about what dishonest users Alice and Bob can do, we can use the key recovery scheme itself for the inner encryption:

1. Using the given scheme with key recovery, Alice produces a ciphertext

   $$c' = < E_{S'}(m') \,||\, E_{pk\_Bob}(S') \,||\, E_{pk\_Tracy}(S') \,||\, bind' >,$$

   where *m’* is her real message, and deletes $E_{pk\_Tracy}(S') \,||\, bind'$ or replaces it by noise.
2. Alice adds a text *info* to this which tells Bob exactly how to process the ciphertext on his side. The result is

   $$m = < info \,||\, E_{S'}(m') \,||\, E_{pk\_Bob}(S') >.$$
3. Alice uses the given scheme with key recovery to superencrypt *m*, resulting in the unsuspicious-looking message

   $$c = < E_S(m) \,||\, E_{pk\_Bob}(S) \,||\, E_{pk\_Tracy}(S) \,||\, bind >.$$

[2] “... by sending noise instead of a third component $<E_{pk\_Tracy}(S)>$ unilateral abuse (i.e., without help of the addressee) is easily possible. This can be prevented in the software of the addressee by a recalculation and validation of C3 $<E_{pk\_Tracy}(S)>$ prior to decryption. However, abuse by colluding of a sender and receiver - through a one-time manipulation of this validation in software - is still easily possible.” [VeTi_97 p. 121]

Note that we are in the correct model: Alice and Bob do not need any prior agreement because Alice can include all the necessary information on what to do in *info*; they do not need another encryption system; and they only perform the simple manipulations allowed according to Section 2.2.

3.2 Failed Attempts to Repair the Idea

We believe that our attack works not only in the model, but also in real life (as pointed out quite clearly in [VeTi_97 p. 121], although using PGP as the inner encryption, forbidden in the model, would actually be simpler in real life).

Nevertheless, let us try to repair the idea by assuming that the recipient *cannot* tamper with his decryption software. However, in this case one does not need binding data either; it is much easier to let Bob's software refuse to decrypt incorrect ciphertexts in a simple key recovery scheme. If $E_{pk\_Tracy}()$ is deterministic, then Bob's software can check the value $E_{pk\_Tracy}(S)$ in c by recomputing it. Otherwise, the entire random string r used by Alice's software to compute $E_{pk\_Tracy}(S)$ can be included in the encrypted message, i.e., as $E_S(r \| m)$. This effectively makes $E_{pk\_Tracy}(S)$ verifiable.[3] Additional verification by fourth parties is not necessary, and would only increase the costs of the system operation.

The next question is: Again assuming that Bob can tamper with his software, can one improve the verification by fourth parties so as to prevent our attack? Obviously, it is not enough to prove additional properties of the outer encryption, but one has to prove that m is not a ciphertext again. As the fourth parties would have to do this without decrypting, it is certainly not efficient. Furthermore, it would require a formal predicate distinguishing encrypted from unencrypted messages, which does not exist: Encrypted messages look as random as compressed data, and if necessary, one can easily encode encrypted messages so that they look like natural language (e.g., “zero zero one zero ...”) and describe the coding rules in clear in *info*.

Note that even if one entirely leaves the model of [VeTi_97], and lets the third parties take part in all message transfers, such verification is not feasible: Firstly, it would require that *all* the traffic, or at least a large fraction of it, be decrypted and inspected by intelligent human censors. This is legally not possible in any moderately free country. (Where people have reason to feel observed by the current government, many will not dare to utter deviating opinions even in private — requiring people to let the government see how they vote, quite unthinkable in democracies, would be a much smaller restriction.) Secondly, Alice could use steganography to conceal her message almost perfectly. This would require Alice and Bob to secretly agree on a steganography scheme in advance, but this is (hopefully) a much more realistic assumption than that any democratic society would establish such a Big Brother organization.

The final question is: Is there any realistic scientific assumption for key recovery? The weakest meaningful requirement is the following: “The binding mechanism does not have to make it arbitrarily difficult to bypass key escrow, but merely as difficult as it is to … or to write a new encryption package to provide comparable security” [WLEB_96 p. 43]. But actually this demonstrates that, strictly speaking, a secure key recovery scheme cannot exist (or, the other way round, that meeting this requirement is trivial[4]):

---

[3] In the specific construction for ElGamal, this means that the secret exponent k used for the encryption is given to Bob. As k is chosen randomly and independently for each session key encryption, this does not give Bob any advantage because he can perfectly simulate the parallel encryption of session keys himself.

[4] [WLEB_96 p. 43] continues “Blaze's work <on misusing Clipper> and the ready availability of encryption software from public sources worldwide <reference to TIS inventory of crypto products of the world> make it clear that this should not be an extraordinarily difficult challenge to meet.”

- There is no need to *build* systems without key recovery. They already exist — for email, file systems, Internet telephony — and are widely available all over the world.
- Effectively using a system without key recovery requires trustworthy key certification for it. But any scheme that guarantees authenticity of messages can be “misused” for this. In particular, if a Public Key Infrastructure guarantees that Bob knows the correct public key for Alice’s digital signatures, Alice can sign her public encryption key and send it to Bob (in a message superencrypted with the system with key recovery). Note that it would not even make a difference if there were also key recovery for signing keys (which makes no sense in other ways either): As long as authentic messages from Alice typically reach Bob, Bob would typically safely obtain Alice's unrecoverable public encryption key.

4 Summary

We have demonstrated how to break key recovery schemes with binding data, as introduced by Verheul et. al., by showing how to modify the well-known superencryption attack on key recovery schemes such that it works without violating any of the assumptions made in [VeTi_97].

The reason why we could do this is not that key recovery schemes with binding data are less secure than the same schemes without binding data in practice. Actually, [VeTi_97] proves that binding data do *not* reduce security. The only reason is that the assumptions about the adversary in [VeTi_97], which justify binding, are slightly more realistic than those typically made for key recovery schemes.

In general, it can be argued that no software key recovery scheme secure against reasonably capable dishonest users can exist. In practice, this means that key recovery might help in eavesdropping on very careless and technically incapable criminals.[5] But it does not really help in tracking organized crime.

Acknowledgments

We thank *Birgit Baum-Waidner* and *Andreas Pfitzmann* for helpful discussions about the problems of key recovery in general, and *Eric Verheul* for a discussion about the scheme attacked here.

Literature

AABB_97	H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier: The Risk of Key Recovery, Key Escrow, and Trusted Third-Party Encryption; The World Wide Web Journal 2/3 (1997) 241-257.
Desm_95	Y. Desmedt: Securing Traceability of Ciphertexts - Towards a Secure Software Key Escrow System; Eurocrypt ’95, LNCS 921, Springer-Verlag, Berlin 1995, 147-157.
FrYu_95	Y. Frankel, M. Yung: Escrow Encryption Systems Visited: Attacks, Analysis and Designs; Crypto ’95, LNCS 963, Springer-Verlag, Berlin 1995, 222-235.
KnPe_96	L. R. Knudsen, T. P. Pedersen: On the difficulty of software key escrow; Eurocrypt ’96, LNCS 1070, Springer-Verlag, Berlin 1996, 237-244.
VeTi_97	E. R. Verheul, H. C. A. van Tilborg: Binding ElGamal: A Fraud-Detectable Alternative to Key-Escrow Proposals; Eurocrypt ’97, LNCS 1233, Springer-Verlag, Berlin 1997, 119-133.
WLEB_96	S. T. Walker, S. B. Lipner, C. M. Ellison, D. M. Balenson: Commercial Key Recovery; Communications of the ACM 39/3 (1996) 41-47.
Zimm_95	Philip R. Zimmermann: The Official PGP User’s Guide; MIT Press, Cambridge 1995.

[5] Although we have serious doubts whether any criminal able to use electronic communication at all could be sufficiently incapable not to install and use something like PGP.