Two Remarks on Public Key Cryptology

Ross Anderson

University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK
Ross.Anderson@cl.cam.ac.uk

In some talks I gave in 1997-98, I put forward two observations on public-key cryptology, concerning forward-secure signatures and compatible weak keys. I did not publish a paper on either of them as they appeared to be rather minor footnotes to public key cryptology. But the work has occasionally been cited (e.g., [5]) and I've been asked to write a permanent record.

1 On the Forward Security of Digital Signatures

At the rump session of Eurocrypt 97, I introduced the idea of a *forward secure digital signature*, on which I elaborated in passing during an invited talk given at the ACMCCS conference later that year. The concept has since been developed further by other researchers [3, 5].

1.1 The basic idea

In 1996, Adam Back floated the idea of a public key cryptosystem with a series of public keys p_i and secret keys s_i that stand in the usual relationship with each other but for which there are updating functions f_i and g_i such that $p_{i+1} = f_i(p_i)$ and $s_{i+1} = g_i(s_i)$ [2]. In this way a single root public key p_0 could be certified, and thereafter the key owner could regularly calculate s_{i+1} and destroy s_i. In this way, the compromise of a private key would not expose traffic encrypted to the key in previous epochs. In 1997 I proposed the obvious extension to digital signatures, in order to prevent the retrospective forgery of messages signed using keys belonging to earlier epochs but without requiring that the public key infrastructure accommodate large numbers of time-limited public keys.

As motivation, note that while Diffie-Hellman key exchange [6] can provide forward security easily in interactive communication, the US Defense Messaging System (DMS) apparently uses transient public keys to provide forward security in offline messaging: when Alice wishes to communicate with Bob, she fetches from a directory server a public key signed with his long-term private key. (DMS is described in [8], and the KEA key agreement algorithm which it uses in [10].) Is it possible to provide such functionality without having to commit to a particular directory access infrastructure?

1.2 Forward secure signature

The forward secure signature method I suggested at Eurocrypt and ACM starts off with the Fiat-Shamir scheme [7]. The basic idea behind this is that a signature

Sig is produced as a combinatorial product of a number of secrets σ_i, which are the roots of public verification values v_i. In its simplest form, one simply multiplies together the secrets σ_i for those values of i for which the i-th bit in the hash of the message to be signed is 1.

$$Sig = \prod_{h_i(M)=1} \sigma_i$$

The secret values might, for example, be the key owner's name, hashed with the integers i, in order to provide a signature scheme that is *identity-based* (that is, the public key is the user's name). In the Fiat-Shamir variant, the trapdoor one-way function which maps σ_i to v_i is squaring modulo a product of two large random primes (to provide provable security) and there is also a random multiplier (so that the secret values σ_i can be resued indefinitely.) This makes the signature $Sig = (r^2, s)$ where

$$s = r \prod_{h_i(M)=1} \sigma_i \pmod{n}$$

My proposal was to have two different features:

- after working out a large enough secret vector σ_i from the public vector v_i, destroy the factors of the modulus
- erase one or more of the σ_i after each signature

Some care is needed in the detailed implementation, such as to prevent any σ_i being reconstructed from signatures using Gaussian elimination; but this is well within the scope of someone skilled in the art.

1.3 Forward secure encryption

The general observation that followed is that any identity based public key cryptosystem or digital signature scheme can be adapted to provide a forward secure system. The trick is that the key owner plays the role of the CA in the identity-based scheme and generates private keys corresponding to the 'identities' of 1, 2, 3, ... or the dates, or whatever else is required.

Thus, for example, we can get forward secure encryption by the obvious adaptation of the Maurer-Yacobi scheme, which employs Diffie-Hellman key exchange with a composite modulus whose factors are known to the CA and with respect to which she can compute discrete logs to obtain the private keys corresponding to given public keys [9].

I also pointed out that 'ID-based' signature schemes are not signature schemes in terms of the usual legal definition of an electronic signature, which imposes a requirement that the signer maintain the means of signature creation under her sole control. Thus perhaps forward security is the real application of this technology.

1.4 Forward versus backward security

There has been some comment recently about whether the protection provided by the above schemes is forward security or backward security. The terms are now controversial (see, e.g., RFC 2828 [12]). Until recently, people used the term 'forward security' loosely to mean a design with the property that the compromise of a current key would have only limited effect, as with Diffie Hellman. How, however, the following distinction is being drawn between backward and forward security.

Backward security may be taken to mean that a compromise of a key now does not necessarily expose old traffic. In traditional systems, it is provided by *key updating*: two or more principals who share a key pass it through a one-way hash function at agreed times: $K_i = h(K_{i-1})$.

Forward security may be taken to mean that a compromise of a key now does not necessarily expose future traffic. Before the advent of public key cryptography, it was achieved by regular rekeying, or in some cases by *autokeying*, in which two or more principals who share a key hash it at agreed times with the messages they have exchanged since the last key change: $K_{+1}i = h(K_i, M_{i1}, M_{i2}, \ldots)$. The point is that if an attacker compromises one of their systems and steals the key, then as soon as they exchange a message which he doesn't observe or guess, security will be recovered. This is fragile in general, as it is easy to lose synchronization: but a variant on the theme is used if EFTPOS terminals in Australia [4]. Diffie-Hellman allows a stronger form of forward security, namely that as soon as a compromised terminal exchanges a message with an uncompromised one, then security can be recovered even if all the traffic is monitored by the opponent.

I believe that this distinction is sensible, and when we use these definitions, the mechanisms described above mainly provide backward security. In some circumstances, they can also provide forward security given suitable surrounding protocols. The simplest way to do this in the case of encryption is to keep all the key material for future use in offline storage. It is also possible to store the private keys (s_i in the terminology of [9]) under keys formed from hash chains running in the appropriate direction. Thus, for example, the keys s_i can be stored encrypted under the keys k_i where $k_j = h(k_{j-1})$, and where each k_j is discarded after use. (This does not stop a compromise of k_i exposing all future keys, so is not forward security.)

In 1997–98 key escrow was a bigger issue than it is now. I remarked that private keys could be stored by the escrow agent too but under the keys e_j where $e_j = h(e_{j+1})$. That way the escrow agent can give the law enforcement agency keys for the back traffic without compromising future keys. This provides forward security of a sort. But in the UK, this question is now moot as recent crypto legislation allows users to revoke public keys at once if the corresponding private keys are confiscated by law enforcement agencies.

2 Compatible Weak Keys

Some programs, such as the support for Microsoft's Crypto API (CAPI) embedded in Windows, verify other programs using embedded public signature verification keys. Sometimes there is a requirement to defeat these mechanisms, such as to circumvent export control or accessory control functions. A naive approach is to find the public key and replace it with another one. The problem here though is that, although all existing modules can be re-signed with the new key, the system may balk at accepting genuine software downloaded later, which could make software upgrades problematic for companies that bought a product which substituted the embedded key. This problem arose in the context of a Cambridge company, nCipher, that needed to defeat CAPI in order to make their products work with Microsoft operating systems.

Another problem is that many people leave uncertified keys in places that are not adequately protected. For example, people leave their PGP keys on their web pages. There appears to be an implicit assumption that if the key is replaced by an attacker, it will be detected quickly as ciphertexts will be received that cannot be decrypted. These observations inspired the following.

Given a public key K and the corresponding private key K^{-1}, a *compatible public key* k is one which will encipher messages so that K^{-1} can decipher them, or verify signatures made by K^{-1}, or both. A *compatible weak key* also has a further private key k^{-1} which will decipher messages enciphered using k, or create signatures that k will verify, or both.

It is easy to create a compatible weak key for RSA encryption. Let the genuine public key be (e, n); if this is replaced (for example, on a public keyring) by (e, nn') where the attacker knows the factors of n', This provides a compatible weak key for some RSA encryption systems. For example, it provides an exploit against the large prime variant of RSA, where the session key must be padded with zeros in its most significant bits, and (I am told) for at least one proprietary system. It does not, as it happens, provide an exploit against PGP version 2 because the software truncates the received key packet to the number of bits declared in the user's private key.

Similarly, there is a possible attack against ElGamal encryption, the basic idea of which is to force the operation into a smooth subgroup [1]. If the encryption key is p, g, y ($= g^x \bmod p$), then it can be replaced with p, g^q, y^q where $p - 1 = qr$ and r is smooth. This does not work against PGP v 5 because of random padding of the key packet's plaintext, which I understand was introduced to stop low-exponent attacks on RSA but kept in PGP version 5 as legacy code.

So here is a new attack on public key cryptosystems. It fails to provide exploits against the most widely fielded applications, but this is a matter of chance rather than of design.

I'm aware of no compatible weak key attacks on signature schemes that work in the general case. As for CAPI, the attack found by nCipher is completely different: it is documented in [11]. I understand from them that in the end it was never necessary to exploit it.

Historical note: I first talked about forward secure signatures sometime in early 1997 at the regular security group meeting at Cambridge University. As far as I can reconstruct from the slides, sections 1.1 and 1.2 were presented at Eurocrypt 97 and section 1.3 was added at ACMCCS. Section 3 was presented at the rump session of Crypto 98, although its first mention in public was at a Cambridge security group meeting in January 1998. Anyone from the audience at any of these events who has more detailed notes is welcome to contact me.

References

1. RJ Anderson, S Vaudenay, "Minding your p's and q's" in *Advances in Cryptology – Asiacrypt 96*, Springer LNCS vol 1163 pp 26–35
2. A Back, "non-interactive forward secrecy", posting to cypherpunks mailing list (6/9/1996), archived at `http://cypherpunks.venona.com/date/1996/09/msg00561.html`
3. A Back, "Non-interactive forward secrecy / identity based crypto ", posting to cypherpunks mailing list (31/5/1998), archived at `http://www.inet-one.com/cypherpunks/dir.1998.05.25-1998.05.31/msg00171.html`
4. HJ Beker, JMK Friend, PW Halliden, ''Simplifying key management in electronic fund transfer point of sale systems", in *Electronics Letters* v 19 (1983) pp 442–443
5. M Bellare, SK Miner, "A Forward-Secure Digital Signature Scheme", in *Advances in Cryptology – Crypto 99* Springer LNCS v 1666 pp 431–448
6. W Diffie, ME Hellman, "New Directions in Cryptography", in *IEEE Transactions on information theory* v 22 no 6 (Nov 76) pp 644–654
7. A Fiat, A Shamir, "How to prove yourself: practical solutions to identification and signature problems", in *Advances in Cryptology — CRYPTO 86*, Springer LNCS v 263 pp 186–194
8. A Kondi, R Davis, "Software Encryption in the DoD", at *NISSC 97* pp 543–554
9. UM Maurer, Y Yacobi, "A Non-interactive Public-Key Distribution System", in *Designs, Codes and Cryptography* v 9 no 3 (Nov 96) pp 305–316
10. National Institute of Standards and Technology, *'SKIPJACK and KEA Algorithms'*, 23/6/98, `http://csrc.nist.gov/encryption/skipjack-kea.htm`
11. A Shamir, N van Someren, "Playing 'Hide and Seek' with Stored Keys", in *Financial Cryptography 1999* pp 118–124
12. R Shirey, *'Internet Security Glossary'*, RFC 2828 (May 2000), at `http://www.es.net/pub/rfcs/rfc2828.txt`