Bandwidth-Optimal Kleptographic Attacks

Adam Young[1] and Moti Yung[2]

[1] Dept. of Computer Science, Columbia University.
[2] CertCo New York, NY, USA.

Abstract. Cryptographic and physical leakage attacks on devices and systems which implement cryptosystems, is an area of much recent activities. One type of attacks are what is called kleptographic attacks which are mounted against black-box cryptosystems. They are issued by and serve solely the designer/manufacturer giving it unique advantage. Kleptographic attacks are capable of leaking the private keys of users securely and subliminally to the manufacturer of the black-box system (based on the availability of public values, such as keys (produced when the system is initiated) or signature/ciphertext values (produced by systems in operation). These attacks provide a very high level of security against reverse-engineering since even if the black-box is successfully reverse-engineered, no information can be obtained that compromises the secrets of the users (thus, the unique advantage of the attacker is retained).
Numerous open questions remain in the area. One issue is that the only key generation procedure with known attack is the RSA/ factoring based PKC, while for Discrete Logarithm based keys attacks are not known. Similarly open, is the existence of bandwidth-optimal leakage attacks, namely attacks on a "single signature" in Discrete Logarithm based signatures (both in the full group and prime order sub-group cases).
In this paper, we solve the above open questions. We develop new attack techniques, which unlike earlier attacks, require *only one value* in order to leak the secret. This gives an attack on modular exponentiation keys. We then show how to implement an attack on ElGamal signature which leaks the private key in each signature, and which requires only 160 bits of smoothness in $p-1$, where p is the common ElGamal prime. The attack utilizes the Newton channel. This channel, however, does not extend to DSA, since DSA operates in a prime order subgroup of Z_p. In the second part of this work, we nevertheless show a subliminal channel attack on DSA that assumes the existence of a small amount of non-volatile memory in the device. This gives a kleptographic attack against DSA that leaks the private key in each signature as well. Non-volatility is only needed to assure the polynomial indistinguishability of the outputs of the devices under attack from that of a normal devices' outputs. We investigate our non-volatility assumption against hardware feasibility (in quite a popular EEPROM devices, used in manufacturing of smart-cards).

Key words: Leakage attacks, subliminal channels, the Newton channel, design methodologies for asymmetric ciphers, kleptographic attacks, attack bandwidth, discrete logarithm based systems, ElGamal, DSA,

Ç.K. Koç, D. Naccache, and C. Paar (Eds.): CHES 2001, LNCS 2162, pp. 235–250, 2001.

tamper-proof hardware designs, trust, public scrutiny, non-volatile memory, hardware technologies: EEPROM, ferroelectric.

1 Introduction

This paper is concerned with attacks by the designer/manufacturer of black-box cryptographic systems. The goal is thus to expose non-trivial leakage attacks which are possible in black-box cipher designs (where the implementation design is not scrutinized, as in tamper-proof hardware devices) but which are not necessarily possible in public designs. Indeed, black box usage of cryptography has been encouraged by governments. The US government, for example, has proposed the use of Capstone, a general purpose crypto-processor which grew out of the Clipper Initiative. Furthermore, since the mid 80's the NSA's Commercial COMSEC Endorsement Program has been active in trying to base cryptography on government designed tamper-proof devices (see [Sch], page 598). In addition, DSA is often included in smart card tokens, which constitutes a black-box environment. DSA was proposed as the Digital Signature Standard in the US [DSS91]. The motivation of this paper is to investigate the possibility of (till now unknown) attacks. These attack involve the design of black-box discrete log based systems with sophisticated trapdoors that are bandwidth-optimal (i.e., leak based on availability of a single value), are hard to detect and immune to reverse-engineering, and at the same time are indistinguishable from the attack-free systems.

Note that it is easy to mount certain attacks on black-box devices, e.g., by fixing (or otherwise specifying) the randomness they use. However, when dealing with black box environments, the risks posed by the real possibility of successful reverse-engineering may out-weigh the benefits of this attack when viewed from the perspective of the malicious designer (being a malicious government, say). For example, if the "random" DSA exponent k is chosen pseudorandomly based on a secret seed rather than being chosen truly at random, then the reverse-engineer will gain as much knowledge as the designer if the seed is learned via reverse-engineering. Furthermore, even if the seed is chosen randomly for each chip, the company responsible for the programming of the chip learns the seeds. On the other hand for example, the attacks we present in this paper are carried out in such a way that even the programmer of the chip gains no knowledge that will help determine the private keys of users. The programmer of the crypto-chip will only learn that a suspicious looking variant of DSA is being implemented. A related, but different, attack on signature schemes which uses weak pseudorandomness has been presented in [BGM97] (in contrast, we use strong randomness/ pseudorandomness in all our attacks, but in combination with a public key and in a different operational setting).

There has been much interest in analyzing cryptosystems with respect to their subliminal leakage. Gus Simmons pioneered much of the work on subliminal channels where the leakage is universal (leaked to everyone) [Si85,Si93,Si94]. Recently, such leakages were suggested not only for leaking information sub-

liminally, but securely (privately) even if the device is later reverse-engineered (increasing the awareness of the need of trust in the manufacturer of black-box devices that looks like they comply with the system's specifications). The basic notions underlying these attacks as well as tools that accomplish them were developed in [YY96,YY97a,YY97b]. Specifically, they introduced the notion of a SETUP attack where a secretly embedded trapdoor (public key) is used to securely leak the secret information out of the cryptosystem. Their attacks are geared specifically towards public key systems and exploit randomness and subliminal channels in key generation, message encryption, and signing. The number of leakages needed for recovery of the secret by the attacker was called the setup "bandwidth." For key generation stage which produces the key and nothing more, an optimal-bandwidth of one value is a must. All the earlier attacks on discrete log based systems like the ElGamal cryptosystem, the ElGamal Digital Signature algorithm and the DSA (Digital Signature Algorithm) leak the private signing key (say) over the course of two (or more) signatures and no bandwidth optimal attack was known. Therefore, key generation attacks on discrete log. systems were not known either (in contrast with RSA/factoring).

The entire issue of optimal bandwidth attacks was open in discrete log based system and we solve it in this work. We first utilize the elegant Newton subliminal channel to mount optimal-bandwidth attacks on discrete logarithm keys and ElGamal signatures. We then apply a new subliminal channel attack on DSA (the first technique does not apply to it). Our second attack requires a limited amount of non-volatile memory in the computing environment. We also investigate the feasibility and attack life time in the required environment in realistic hardware devices employing EEPROM and the emerging ferroelectric technologies.

Organization: Next we recall and present the basic definitions of the notions and systems we use. Section 3 presents the attack based on the Newton channel discrete log cryptosystems. Sections 4 and 5 then give and analyze the attack on the DSA scheme. The conclusion is in Section 6, and the Appendix presents detailed hardware background explaining available implementations of some of our conditions in existing hardware technologies.

2 Definitions

Our attacks utilize the notion of what is called a SETUP attack (Secretly Embedded Trapdoor with Universal Protection). The following is the definition of a (regular) setup [YY97a]:

Definition 1. *Assume that C is a black-box cryptosystem with a publicly known specification. A SETUP mechanism is an algorithmic modification made to C to get C' such that:*

1. *The input of C' agrees with the public specifications of the input of C.*
2. *C' computes efficiently using the attacker's public encryption function E (and possibly other functions as well), contained within C'.*

3. *The attacker's private decryption function D is not contained within C' and is known only by the attacker.*
4. *The output of C' agrees with the public specifications of the output of C. At the same time, it contains published bits (of the user's secret key) which are easily derivable by the attacker (the output can be generated during key-generation or during system operation like message sending).*
5. *Furthermore, the output of C and C' are polynomially indistinguishable to everyone except the attacker.*
6. *After the discovery of the specifics of the setup algorithm and after discovering its presence in the implementation (e.g., reverse-engineering of hardware tamper-proof device), users (except the attacker) cannot determine past (or future) keys.*

Observe that the above definition does not quantify the number of invocations of C' for which the SETUP attack is carried out. Hence, it is implicitly assumed that for all invocations of C', the output contains the published bits of the user's secret key. In this work we change this quantification from being unbounded to being polynomially bounded in some security parameter k (typically, the same security parameter as in the underlying cryptosystem). This small change of explicit bound is merely motivated by the reality of hardware devices, with finite ability to keep/ change a certain state (as a property of the underlying technology – we will see an example later). Informally the attack works as follows, the attacker chooses some polynomial $poly'$ (in k) and implements C'. For the first $poly'$ invocations of C', the SETUP attack will be in effect. The user then chooses his or her own polynomial $poly$, and runs C' that many times. Only if $poly > poly'$ are there invocations of C' for which the SETUP attack is not carried out (i.e., C' behaves honestly). In this case the last $poly - poly'$ invocations of C' are identical to C. In our attack on DSA, we show that if $poly'$ is large enough (which is achievable in practice), then C' will have to be invoked polynomially many times to reach the point at which it behaves identically to C (in the appendix we show how this can easily be 14 years under a reasonable pace assumption, by which time the technology is likely to become obsolete).

Below we give a definition of a SETUP attack to reflect this idea of boundedness.

Definition 2. *Assume that C is a black-box cryptosystem with a publicly known specification. A* (poly)-bounded *SETUP mechanism is an algorithmic modification made to C to get C' such that it has the six properties of SETUP and in addition it has the following property:*
7. The SETUP attack is carried out a polynomial number of times in k (where k is the security parameter of the underlying cryptosystem), after which C' behaves identically to C.

Signature Schemes:

The following is a review of the ElGamal digital signature algorithm [ElG85]. Let p be a large prime, and let $g \in Z_p$ be an element with order $p-1$. The signing private key is $x \in_R Z_{p-1}$, and the public signature verification key is

$y = g^x \ mod \ p$. Let H be a one-way hash function. To sign an arbitrary message m, the following algorithm is performed:

1. $k \in_R Z_{p-1}$
2. $a = g^k \ mod \ p$
3. $b = k^{-1}(H(m) - xa) \ mod \ p - 1$
4. output the signature (a, b)

To verify (a, b) the verifier makes sure that $y^a a^b = g^{H(m)} \ mod \ p$.

We will review the Digital Signature Algorithm (DSA). Let p be a large prime number such that $q \mid p - 1$ where q concretely is chosen to be a 160 bit prime number and p is standardized to some concrete range (512-2048 bits length) as well. Let g be an element in Z_p with order q. The signing private key is x where $x \in_R Z_q$, and the public verification key is $y = g^x \ mod \ p$. Let SHA denote the Secure Hash Algorithm. To sign the message m, we compute:

1. $k \in_R Z_q$
2. $a = (g^k \ mod \ p) \ mod \ q$
3. $b = k^{-1}(SHA(m) + xa) \ mod \ q$
4. output the signature (a, b)

To verify (a, b) the verifier makes sure that $a = (g^{SHA(m)/b} y^{a/b} \ mod \ p) \ mod \ q$.

Finally, we will now introduce the underlying cryptographic assumption which is utilized in both attacks in this paper. This assumption is based on the Diffie-Hellman (DH) assumption [DH76], but adds additional hiding of the secret. Here g and the prime p are public, and v, v' which divide $p - 1$ are also public. We will refer to it as the Diffie-Hellman Plus Sum (DH-PS) assumption.

Diffie-Hellman Plus Sum Assumption: Let $A = g^a \ mod \ p$, $B = g^b \ mod \ p$, and $c = a+b \ mod \ v$, where $v \mid p-1$. It is intractable to compute $(g^{ab} \ mod \ p) \ mod \ v'$ where $v' \mid p - 1$, and $|v'| \geq M$ (concretely M is 160), given A, B, and c.

Here $|v'|$ denotes the bit length of v'. Clearly, if we can solve DH, then we can solve the DH-PS assumption and the DH-PS is randomly self reducible (namely, we can randomize the input instance and if the randomized new instance is solvable, we can translate the result to the result of the original input instance).

3 SETUP Attacks Based on the Newton Channel

We start by showing optimal attacks on cryptosystems that operate in all of Z_p (ElGamal type), rather than in a prime order subgroup. Our SETUP attack utilizes the the Newton Channel and leaks the private key in each and every signature. The Newton Channel was given in [AVPN].

3.1 Review of the Newton Channel

Let $p = qm + 1$ be prime, and let q be prime. Furthermore, assume that m is smooth, and that g generates Z_p. For security it is assumed that computing discrete logs in the group generated by g^m is hard. Let c be the covert message that is to be displayed. To display c in an exponentiation mod p using base g, a value $k'\ mod\ (p-1)/m$ is chosen randomly, and we solve for k in $k = c + k'm\ mod\ p - 1$. Hence,

$$k \equiv c\ mod\ m$$

The user then publishes $r = g^k\ mod\ p$, as in any discrete log based cryptosystem (such as the ElGamal digital signature scheme). The recipient, who can be anyone who decides to recover c, can recover c as follows. The recipient solves for z in the equation,

$$(g^q)^z \equiv r^q\ mod\ p$$

This can be done since the order of the subgroup of Z_p generated by g^q is smooth. Let B be the largest prime in m (i.e., its smoothness). Using Pohlig-Hellman [Poh78] and Polard's Rho [Pol78], this requires time $O(B^{1/2})$. It then follows that,

$$c \equiv z\ mod\ m$$

The clever Newton Channel was also modified to become narrowcast as opposed to the broadcast channel given above. This is done by replacing q with two different primes q_1 and q_2, and having the sender and receiver a priori secretly share the signing private key mod q_2 and having the sender keep the signing key mod q_1 private. This however, requires a more specialized form for the factorization of $p - 1$, and may result in reducing the security of the underlying system. In the SETUP we describe below, no such a priori secret exchange is required, and this specialized form for $p - 1$ is not needed. Thus, the SETUP attack can be utilized securely and subliminally under the observance of a warden, as in the case of Simmons original Prisoner's Problem, *without* requiring that the prisoners exchange a secret before going to prison.

3.2 Setting up a Discrete Log Based Key Generation

It is now not hard to add a setup attack to the Newton Channel to leak the private exponent x in $y = g^x\ mod\ p$ where g generates Z_p and $p = mm'q + 1$. Here m is smooth and q is a prime which is greater than or equal to 160 bits in length. m' can be any value. The attack is mounted by computing $c = E(seed)$, where E is a public key encryption algorithm. E can be an elliptic curve

cryptosystem which outputs 310 bit ciphertexts. Hence, m must be 310 bits in this case. Provided $c < m$, the value for *seed* is used in the attack. Rather than choosing k' randomly as in the Newton Channel, it can be chosen by applying a pseudorandom generator to the value *seed*. The range of the pseudorandom generator is $Z_{m'q}$. k is computed in the same way as in the Newton Channel. Therefore, anyone in possession of the private key corresponding to E can recover the value for *seed* and reconstruct k' and hence k.

Theorem 1 *Assuming there is a smooth number within the factorization of* $p-1$ *which is large enough to be greater than the size of public key ciphertexts, there exists a SETUP attack against any public exponentiation modulo* p *under the Diffie-Hellman plus Sum assumption.*

3.3 Adding a SETUP Mechanism on Top of the Newton Channel

Let q be prime, and let $p = mm'q + 1$. We insist that m is smooth and even and 160 bits in length. We do not insist on any particular form for m'. We will make the simplifying assumption that q is 160 bits, though many different configurations on the sizes of m and q are possible. We only require that $q < m$. Let x' *mod* q be the attacker's private key, and let $y' = g^{mm'x'}$ *mod* p be the attackers corresponding public key (which is not published). Let $x \in_R Z_{p-1}$ be the unwary user's private key, and let $y = g^x$ *mod* p be the corresponding public key. The attack aims to securely and subliminally leak x in the sense of a SETUP to the attacker via $r = g^k$ *mod* p (as in the Newton Channel), which is output by the device.

Assume that y' has been placed in the discrete log cryptosystem device that is to be SETUP. The attack is mounted as follows. The device chooses $R \in_R Z_q$. The device then solves for c in,

$$c = R + x \ mod \ q$$

Now, unlike in the Newton Channel, k' is *not* chosen randomly. Instead, the device computes k' pseudorandomly based on y' and c. To be more specific, the device solves for k' as follows.

$$k' = H(y'^R \ mod \ p)$$

where H is a public pseudorandom function [GGM86] which uses a secret seed that only the attacker and the device knows. We assume that the range of H is $Z_{m'q}$. The device then computes $k = c + k'm \ mod \ p-1$. The device outputs $r = g^k$ *mod* p as in the Newton Channel. This value can, for instance be the first value in the pair of values which constitute and ElGamal digital signature.

The attacker can recover x from r as follows. The attacker recovers c in the same way as everyone can using the Newton Channel. The attacker then computes $t = g^c y^{-1}$ *mod* p. Note that $t = g^R$ *mod* p. The attacker then solves for k' as follows.

$$k' = H(t^{mm'x'} \ mod \ p)$$

k is then recovered by computing $k = c + k'm \ mod \ p - 1$. In most (if not all) digital signature algorithms, such as ElGamal, knowledge of k implies knowledge of x.

3.4 Security

Note that the overall security of x is inherently reduced for the users of the system overall, due to the existing smoothness in $p - 1$. If we suppose in the worst case that $(p - 1)/q$ is entirely smooth, then the users really only posses private keys of the form $x \ mod \ q$. So, it is this private key that we will show is intractable to recover without x'. Hence, to show security, we must show that it is intractable to recover $x \ mod \ q$ without x'. To show that it is a SETUP, we must show that the chosen r is polynomially indistinguishable from normally constructed (chosen) values r given $x \ mod \ q$ and not given the secret seed to H.

Claim 1 *It is intractable to recover $x \ mod \ q$ without x' given r, y', $x \ mod \ m'$, and the secret seed to H, assuming that the Diffie-Hellman plus Sum assumption holds.*

Proof. Since $p - 1$ has the requisite amount of smoothness, c can be computed efficiently from r by anyone. Thus, we know k iff we know k', because k is the Chinese Remainder of c and k'. Since k is the "randomly chosen" secret exponent used in the signature which is output (i.e., the secret exponent used to construct the ElGamal signature pair), k is known iff x is known. It follows that k' is known iff $x \ mod \ q$ is known, since we are given $x \ mod \ m'$. From c we then compute $t = g^R = g^c y^{-1} \ mod \ p$. Now, in the absence of the application of H in the attack, we know k' iff we can solve the DH-PS problem with $v = q$ and $v' = (p - 1)/q$, since we know $t = g^R \ mod \ p$, $y' = g^{x'} \ mod \ p$, and $c = R + x \ mod \ q$. Thus, adding the use of H in no way helps in recovering x without x'. Hence, for secrecy of $x \ mod \ q$, knowledge of the secret seed to H is thus superfluous. QED.

If $x \ mod \ m'$ is not given away, then the security of the system still holds. It follows that even if the device is reverse-engineered at a later time and y' is found, this does not help the reverse-engineer figure out x. Hence, property (6) in the definition of a SETUP holds. We note that we had originally tried to reduce the security of this system to that of DSA itself. The idea was to make c the DSA "signature equation", which has the same effect as above: it hides $x \ mod \ q$ using a randomly chosen value mod q. The problem is that all signature equations include a variable which is a commitment of the randomly chosen signature exponent, which in the case of our attack, hasn't even been computed yet (we *must* Chinese Remainder k' with c to get it).

Claim 2 *the random variables: k in the attack and k as computed normally are polynomially indistinguishable given x and y', but not given the secret seed to H.*

Proof. Everyone knows c for each signature. Since c and x are known, R can be found from the equation $c = R + x \ mod \ q$. Once R is obtained, the quantity $(y'^R \ mod \ p) \ mod \ (p-1)/q$ can be computed. Recall that in a given signature which has been set up, this is the preimage under H of k'. However, without knowing the secret seed to the pseudorandom function H, it follows from the definition of a pseudorandom function that the output of H (in this case k') is indistinguishable from randomness (by standard arguments a distinguisher for the function can be constructed from a distinguisher for the random variables). Since c is truly random, and since k' is pseudorandom, when they are Chinese Remaindered, the resulting k is pseudorandom and the same argument follows. QED.

If the exponents used to compute the signatures in each case are indistinguishable (as random variables), then the presence of attacks in the device are also indistinguishable since from the perspective of a user who knows his own private key, k is the only information conveyed to the user by the device (it can be recovered using the user's own x). It follows that property (5) of a being a SETUP is satisfied. Note that if each device is given a unique secret seed, then reverse engineering one device does not help in determining whether another device is contaminated (in other words: under attack) or not. Also, whether or not this seed is known, forward security of x holds due to the use of DH-PS in the attack. Properties 1 through 4 of a SETUP hold for this system. So, we have therefore shown the following.

Theorem 2 *Assuming at least M ($M = 160$) bits of smoothness in $p-1$, there exists a SETUP attack against ElGamal (and its variants) that leaks the private key in each digital signature, assuming the security of Diffie-Hellman plus Sum.*

4 Attack on Subgroup Based Signature Schemes

It was observed in [AVPN] that DSA does not support the Newton Channel. This is because all of the users of the system use a value g which generates a prime order subgroup of Z_p whereas the existence of the Newton Channel requires g whose order has some smoothness. The question therefore remains whether or not an optimal bandwidth SETUP attack exists against DSA and its variants (e.g., Nyberg Rueppel [NR94]). In this section we answer this question in the affirmative.

The attack below relies on two specific realistic conditions. First, we assume that each specific cryptographic black-box contains within it a unique private random identifier string. This requirement can be practically met using a keyed hash function during the programming of the crypto device (knowing the key for the hash function does not compromise the DSA private keys of users, however). Note that each Capstone chip has a unique device identifier[1].

The second requirement is that each black-box device has poly-sized non-volatile memory which can be read and written to. This can be realized using

[1] which may differ from another identifier stored in Capstone's E^2PROM

Electrically Erasable Programmable Read-Only Memory (E^2PROM), for instance.

4.1 System Setup

To mount the attack, the designer generates a private key $x' \in_R Z_q$ and places the corresponding public key $y' = g^{x'} \ mod \ p$ in the device. A portion of the non-volatile memory in the device will be used to store a counter cnt which is initially zero. This counter will be incremented by one for each signature that is output by the device. Let MAX denote the maximum value of cnt. Also, let ID denote the unique (cryptographically secure) identifier for the black-box. Hence, each device initially contains the triple (y', cnt, ID) where ID varies from device to device.

4.2 Signing and Verifying

To sign the message m, the device does not choose the DSA exponent k randomly, but rather chooses it pseudorandomly. Here x, y, g, p, q are as in DSA. The following is the SETUP version of the DSA signing algorithm:

1. read cnt from non-volatile memory
2. if $cnt \geq MAX$ then
3. $\quad k \in_R Z_q$
4. else
5. $\quad B = H(ID, cnt)$
6. $\quad k' = B - x \ mod \ q$
7. $\quad k = (y'^{k'} \ mod \ p) \ mod \ q$
8. $\quad$ write $cnt = cnt + 1$ to non-volatile memory
9. $a = (g^k \ mod \ p) \ mod \ q$
10. $b = k^{-1}(SHA(m) + xa) \ mod \ q$
11. output (a, b) as the signature on m

Here H is a publicly specified pseudorandom function, and ID is used as the secret seed to it. We assume that the range of H is Z_q. The signature is verified as in normal DSA. The intuition behind this attack is that B, which would typically be displayed through a subliminal channel, is in fact not displayed at all since it is already known to the malicious designer.

4.3 Recovering the Signing Private Key

Given x' and the list of device IDs, the signing private key x can be recovered from (y, m, a, b) as follows:

1. for each device identifier ID do:
2. $\quad$ for $i = 0$ to MAX do:
3. $\quad\quad B = H(ID, i)$

4. $k = ((g^B y^{-1})^{x'} \ mod \ p) \ mod \ q$
5. if $a = (g^k \ mod \ p) \ mod \ q$ (i.e. the DSA signature check passes) then
6. output $x = (bk - SHA(m))a^{-1} \ mod \ q$ and halt with TRUE
7. halt with FALSE

The private signing key is found iff the device halts with TRUE. Note that if MAX is really large, then an incremental search algorithm may be preferred over the above, depending upon the number of devices in existence and on how much is known about the particular device that was used to compute (a, b) (e.g., starting from an expected number of signatures at this point of time and searching up and down incrementally). Note that even the programmer of the signing algorithm cannot recover x, since the programmer does not know x', only the person who generates y' knows x' (which is presumably the person who gave the programmer the code to burn into the chip). Note that the user can re-key y anytime without affecting the attack. See the appendix for details on how the counter can be implemented using existing non-volatile semiconductor memory technologies.

5 Security

Claim 3 *It is intractable to recover x without x' given y', i, and the seed ID to H, assuming that the Diffie-Hellman plus Sum assumption holds.*

Proof. Given ID and i, the value for B is known. Thus, the value $g^{k'} \ mod \ p$ is known, since $g^B y^{-1} = g^{k'} \ mod \ p$. Now, due to the fact that k is used as the randomly chosen DSA signature exponent for the signature being constructed, k is known iff x is known (the signature which is output by the device is employed). It remains to show that k (and therefore x) can be found iff the Diffie-Hellman plus Sum assumption does not hold. Since B is pseudorandom, k' is pseudorandom, and hence finding k is exactly the Diffie-Hellman plus Sum problem with $v = v' = q$. QED.

Thus, even the reverse-engineer who obtains ID, y', and the secret seed to H cannot determine past or future private keys x. Hence, property (6) of a SETUP holds for this attack.

Claim 4 *The values for k which are used in the above attack are polynomially indistinguishable from the values k chosen in normal DSA signatures, given y', x, and i, but not given the secret seed ID to H.*

Proof. Since H is a privately seeded pseudorandom function using seed ID, for the first MAX invocations of the device B is chosen pseudorandomly and is indistinguishable from random choices. Since B is pseudorandom, it follows that k' is pseudorandom mod q. This means that k results from pseudorandomly chosen values from Z_p which are then reduced mod q, whereas the original k is generated similarly but with random elements in Z_p. If the spaces are polynomial-time distinguishable, by standard arguments one can contradict the pseudorandomness

of H. It follows that the values for k in the first MAX invocations are indistinguishable from random choices. For the remaining invocations, k is chosen as in DSA itself. QED.

Therefore, the device's behavior is indistinguishable from an uncontaminated device, even for the user who knows x, and hence property (5) of a SETUP is therefore met. By placing unique ID's in each device, the reverse-engineer cannot distinguish contaminated devices from uncontaminated ones without individually reverse-engineering them all. Properties 1-4, and 7 hold for this attack. We have therefore shown the following.

Theorem 3 *There exists a poly-bounded SETUP attack against DSA which leaks the private key of the user in each signature based on the Diffie-Hellman plus Sum assumption in a prime order subgroup.*

We note that In the attack above, it would be possible to eliminate the need for writable non-volatile memory if a reliable source of time is available. If the time counter is never reset, and it has sufficient resolution that the same time value is never used for more than one signature, and the attacker can guess the time of signing well enough that it is practical to try all possibilities, then the counter can be eliminated and time can be used as the input to the pseudorandom function. Whether a counter or a timer is more practical depends on the application and the setting.

6 Conclusion

We showed how to use the Newton Channel to implement an optimal SETUP attack against ElGamal Signatures assuming 160 bits of smoothness in $p-1$, based on the Diffie-Hellman plus Sum problem. The notion of a poly-bounded SETUP attack was introduced and an optimal SETUP attack on DSA was presented which securely and subliminally leaks the DSA private key to the implementor in each signature. Hence, in the attack on DSA it was shown that explicit subliminal channels are not needed at all to effectively leak private DSA keys at an optimal bandwidth. These results imply that a single signature can leak a secret securely if a manufacturer attacks a black-box implementation. A cryptographic assumption of perhaps independent interest is utilized.

References

[AVPN] R. Anderson, S. Vaudenay, B. Preneel, K. Nyberg. The Newton Channel. In *Workshop on Information Hiding*, Isaac Newton Institute, 1996. (also downloaded from Ross Anderson's homepage).

[BGM97] M. Bellare, S. Goldwasser and D. Micciancio. Pseudo-Random Number Generation within Cryptographic Algorithms: the DSS Case. In *Advances in Cryptology—CRYPTO '97*, Springer-Verlag.

[DH76] W. Diffie, M. Hellman. New Directions in Cryptography. In volume IT-22, n. 6 of *IEEE Transactions on Information Theory*, pages 644–654, Nov. 1976.

[DSS91] Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). In volume 56, n. 169 of *Federal Register*, pages 42980–42982, 1991.

[ElG85] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology—CRYPTO '84*, pages 10–18, 1985. Springer-Verlag.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali, How to Construct Random Functions. In *Journal of the ACM*, 33(4), pages 210–217, 1986.

[LMS] J. Lacy, D. Mitchell, W. Schell. CryptoLib: Cryptography in Software. In *Proceedings of the IV UNIX Security Symposium*, USENIX Association.

[NR94] K. Nyberg, R. Rueppel. Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. In *Advances in Cryptology—Eurocrypt '94*, pages 182–193, 1994. Springer-Verlag.

[Poh78] S. C. Pohlig. An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. In *IEEE Transactions on Information Theory*, v. 24, n. 1, pages 106–110, 1978.

[Pol78] J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). In *Mathematics of Computation*, v. 32, n. 143, pages 918–924, 1978.

[Sch] B. Schneier. Applied Cryptography, 1994. John Wiley and Sons, Inc.

[Sc91] C. Schnorr. Efficient Signature Generation by Smart Cards. In *Journal of Cryptology*, v. 4, pages 161–174, 1991.

[Si85] G. J. Simmons. The subliminal Channel and Digital Signatures. In *Advances in Cryptology—Eurocrypt '84*, pages 51–57, 1985.

[Si93] G. J. Simmons. Subliminal Communication is Easy Using the DSA. In *Advances in Cryptology—Eurocrypt '93*, 1993.

[Si94] G. J. Simmons. Subliminal Channels: past and present. In *European Tra. on Telecommunications* V. 5, 1994, pages 459–473, 1994.

[YY96] A. Young, M. Yung. The Dark Side of Black-Box Cryptography. In *Advances in Cryptology—CRYPTO '96*, pages 89–103, Springer-Verlag.

[YY97a] A. Young, M. Yung. Kleptography: Using Cryptography against Cryptography. In *Advances in Cryptology—Eurocrypt '97*, pages 62–74. Springer-Verlag.

[YY97b] A. Young, M. Yung. The Prevalence of Kleptographic Attacks on Discrete-Log Based Cryptosystems. In *Advances in Cryptology—CRYPTO '97*, Springer-Verlag.

A Appendix: Implementing the Counter

We will now describe ways of implementing the non-volatile counter using existing technologies, taking care to observe the precise physical operating limitations of these technologies. In particular we describe how to implement the counter on the ST19SF64 chip from ST Microelectronics. We conclude with a description of how the counter can be greatly simplified using emerging (ferroelectric) technologies.

A.1 Background Information on EEPROMs

The predecessor of Electrically Erasable Programmable Read Only Memories (EEPROM) memories was Erasable PROM memories (EPROM) which require the use of UV light for erasure. Though EPROM memories can be rewritten a number of times, a quartz crystal window is needed in the chip to permit erasure, and erasing typically requires 20 minutes or so. The first available EEPROM chips also allowed multiple writes, but unlike EPROMs, the memory could be erased electrically using around 20 volts or so. This however required a separate pin on the chip for the high programming voltage, whereas only 5 volts (the standard operating voltage) was needed for the read operation. Eventually the technology advanced to where only 5 volts were needed for reads and writes. These chips contain voltage amplifiers internally to perform the erase operation.

It is a thesis of this paper that this 5 volt (and lower) EEPROM technology marked a major turning point in the level of trust that must be placed in the manufacturers of cryptographic processors, whether the processors contain non-volatile memory or not. The reason for this is that for the first time stateless tamper-proof microprocessors became indistinguishable from tamper-proof microprocessors containing EEPROM, since no crystal window is needed, and since the operating voltages are the same.

EEPROMs have two major operating characteristics: durability and data retention. Durability refers to the number of times in which a given byte can be erased and written to, and data retention refers to how long a byte can reliably store its value after it is written to. Modern EEPROM memories typically have an endurance of 10^5 and a data retention value of 10 years or so. Note that with these characteristics, in theory a byte can be used for more than 10 years, provided that it is not written to more than 100,000 times, and provided that no more than 10 years passes between each write (in many cases the retention has to do with the discharging of a capacitive layer in the memory cell). These operating characteristics imply that it is not possible to simply utilize 4 bytes of EEPROM as a 64 bit counter, since the lower order byte is not durable enough to handle that many writes.

The reasons for these limitations have to do with the device physics of modern EEPROMs. We will now briefly summarize why these limitations exist. Modern EEPROMs are based on a stored charge concept in which the presence or absence of a stored charge in a MOSFET transistor (typically in a "floating" gate which is isolated by SiO_2) affects the flow of electrons from the drain to the source leads. Another technology (SNOS) utilizes charge trapping material instead of a floating gate. The presence or absence of (a significant amount of current) between these leads indicates a binary 0 or 1, and this current is controlled by the electric field given off the stored charge if charge is present. A number of methods are used to inject and remove the stored electrons in the transistor, the most prominent being quantum mechanical tunneling, and hot electron injection.

The factors affecting endurance are tunnel oxide breakdown, gate oxide breakdown, and trap-up. The first two cause short circuits in the device, thus rendering it unable to store charge. Trap-up refers to electrons being trapped in the tun-

neling insulator, thus weakening the injection fields and therefore not allowing enough charge to get to the stored charge area during programming. For floating gate devices, there is no intrinsic retention problem (and is limited only by device defects). Clearly, full testing is not possible since the tests would be concluded long after the competitive lifetime of the chip. High temperature tests are thus performed, and most retention failures are actually endurance failures. The retention characteristics are different for charge trapping devices, though these are typically only used in military applications which require reliable operation in radioactive environments (e.g., to insure reliable missile guidance systems in fallout).

A.2 An Attack Using the ST19SF64

Below are the specifications for the ST19SF64 CMOS Smartcard MCU chip by ST Microelectronics, with 64k EEPROM, 32k user ROM, and 960 bytes RAM.

Byte write time = 1 milisecond
Data Retention = 10 years
Automatic write operation with internal control timer
$V_{CC} = 5$ Volts (or 3 volts)
Endurance = 100,000 erase/write cycles

Note that the read operation requires on the order of nanoseconds.

Typically, cryptoprocessors utilize a portion of E^2PROM for the cryptocode, so we will assume in our attack that 32k of E^2PROM is reserved for cryptographic operations and that 32k are available to implement the counter *cnt*. We assume that the secret cryptographic device identifier ID, and the attacker's public key y' are stored in the 32k of E^2PROM along with the cryptographic code.

A.3 Implementing the Non-volatile Counter

If we were to utilize, say 4 bytes of non-volatile memory for a counter *cnt* that is incremented from zero we will not get very far since only 100,000 writes can be made reliably. We thus need to design a counter that can exceed 100,000 utilizing the 32k available bytes. First observe that the counter value need not be incremented by one, since it is simply used as an argument to a random oracle H. Hence, all that is needed is a polynomial number of unique values for *cnt*. Using this observation, a counter permitting 2^{30} different values can be synthesized as follows.

The counter *cnt* is the entire 32k bytes. We divide the 32k byte memory space into 16k words, each of which is 2 bytes. Initially, every word is zero. To increment the counter, we read in the words from memory until the least significant word which is not all binary 1's is found (if any). We then add 1 to this value if found. Note that each word can only be incremented at most $2^{16} - 1$ times. It follows that each byte is only written to at most $2^{16} - 1$ times (which is

less than 100,000 as required). The counter can no longer be incremented when all of the bits in the counter are 1. It follows that there are $2^{14} * 2^{16} = 2^{30}$ different possible values for *cnt*. This implementation of *cnt* requires that all 32k bytes of *cnt* be read in the computation of B using the random oracle H. Since reads require on the order of nanoseconds, this is still very fast in comparison to the time required to compute the signature.

A.4 Operating Statistics of the Attack

We were unable to find benchmarks for the time required to compute a DSA signature on a dedicated crypto-processor. So, we will cite the time required to compute a DSA signature in software on a SPARC II using CryptoLib. The time required in this case is 430 milliseconds where $|p| = 768$ bits [LMS]. Note that Cryptolib uses some of the best algorithms to do modular exponentiation, including Montgomery Reduction, Vector Addition Chains, and Karatsuba multiplication. Below we give some of the characteristics of our attack in this setting:

SPARC II CryptoLib DSA signing time = 430 ms
time to read *cnt* (i.e., time to read all 32k bytes) = 4.92 ms
time to update $cnt \leq 4.92 + 2 = 6.92$ ms
total non-volatile memory based overhead ≤ 11.84 ms
number of signatures which are SETUP $= 2^{30}$
time required to exhaust $cnt = 2^{30} * 430\ ms > 14$ years

It follows from the above that the time required to mount the attack can be "absorbed" by the time required to compute the DSA signature. Measures should be taken to insure that the signing time is the same whether or not all values for *cnt* have been exhausted, to avoid detection of the attack.