Anonymous Revocable Attestation

Carl Ellison and Stephen Farrell

This TCD CS technical report exists purely so as to allow some current work to reference an idea developed by the authors in 2002, but which was never published. Other than this paragraph, the document is exactly as it was back then. As of June 2006, Carl can be contacted at cme@acm.org and Stephen at stephen.farrell@cs.tcd.ie

Abstract. There are hardware devices that need to attest by digital signature to their belonging to a particular class but that need, for reasons of privacy, to operate anonymously. Should such a device be compromised, its attestation key should be revoked. Through conventional techniques it is difficult to achieve both anonymity and revocation, although either is easily achievable by itself. In this paper we present a new mechanism, *attestation pools*, to accomplish both goals simultaneously. We also give equations for guiding an implementer in choosing parameters the use of attestation pools.

Keywords. TCPA, TPM, attestation, public-key cryptography, digital signatures, tamper resistance, anonymity, key revocation

1 Introduction

There exist designs for hardware devices, such as the TCPA[8] Trusted Platform Module (TPM), and certain sensors that were to be deployed in support of the Comprehensive Test Ban Treaty (CTBT)[4], that are intended to hold cryptographic keys and other values, and protect them from tampering. Some of these values (such as keys) are to be kept secret. Others are to be reported to various remote parties. When those values are reported, they must be protected from tampering during communication, typically by a digital signature using a cryptographic key kept within the device. In this paper, we call this signed reporting operation **attestation**.

The **attestation key** is a digital signature key that must be certified and the certificate for that key needs to be available to the remote party. In the case of an attestation device contained in a human user's computer, that certificate is intended to identify the class of device that holds the key but not give an identifier for the individual device, much less an ID of the computer of which the device is a part, much less an ID of the human who is using that computer. Those other forms of ID are for other mechanisms to establish, presumably mechanisms under the user's control. It is assumed that the hardware device attestation considered here is not under the user's control and therefore that the

user must be protected from any possible privacy violations as a result of this attestation. We therefore require that attestation be **anonymous** when used in a normal computer.

In the event an attestation key is compromised, by cryptanalysis or physical penetration of the device, it is strongly desirable that the key be **revocable**. Otherwise, a compromised key might be used in a software simulation of the hardware device to attest to false values, while presenting valid credentials and digital signatures. Such a simulation would violate the prime purpose of the attestation key and its certificate: to prove to the remote challenger the type of device doing the attestation, because that hardware device could be trusted to tell the truth about values being reported.

Section 2 strictly defines the attestation protocol that we address in this paper. Section 3 gives a summary of conventional methods to achieve attestation, with or without meeting these two additional goals, with a tabular summary in section 3.5, showing that none of the current methods achieves these two goals. Section 4 describes a new method to achieve anonymous revocable attestation. Section 5 discusses the proper selection of parameters for this new method, to fit the requirements of an intended implementation. Section 6 discusses the problem of detecting compromise of this hardware device and shows that to be a still unsolved problem. Finally, in section 7 we give our conclusions.

2 Attestation Protocol

We assume a basic two party protocol. We label those parties the **challenger** and the **prover**. The challenger is some party connected by communication channel to a computer containing the hardware device. The prover is that hardware device. That communication channel might include multiple computers along its path.

The challenger wants some information held within the prover and requests it. The prover might know ahead of time what the challenger wants to know, but for replay prevention, the challenger should provide a nonce to be returned by the prover along with the requested data and therefore needs to send the first message in the protocol. The protocol is completed with a signed response from the prover to the challenger, including under that signature both the desired data and the freshness data (e.g., the nonce the challenger had sent).

One might design an attestation protocol to use synchronized clocks between the challenger and the prover to ensure freshness. In that case, the first message may be skipped if the prover knows what data the challenger wants to see and we get a single-message "protocol". Of course, use of a nonce for freshness has the advantage of not requiring trust in clock synchronization.

We define this protocol as being between the challenger and the hardware device, without reference to the computer platform containing that hardware device or one alleging to contain that device.

In some cases, one wants to attest to some characteristic not about the hardware device but rather about the platform to which it is attached. Such a use of attestation is subject to a slave-platform attack. For example, one might have a challenger, A, connected by TCP/IP to computer B, which claims to have an attestation device. When A sends a challenge to B, B could then pass that challenge along to computer C, which really does have an attestation device, D, get device D to respond to the challenge and then return that challenge to B for it to return to A. The mechanisms to prove the binding between the attestation device, D, and the communicating partner, B, are outside the scope of this paper. We assume that if attestation is being used in this manner, such a mechanism (and corresponding protocol enhancement) is in place.

3 Methods of Attestation

There are at least four different ways to achieve attestation with public key technology, each partially achieving the goals stated in section 1.

3.1 Individual Key Pair

One can give each device its own attestation signature key pair. When data are to be reported by the device, those data can be signed by this key. The key would be certified by some competent party (most likely the platform manufacturer) to have been safely embedded in the given device. An attestation of some protected value(s) would then consist of a message holding that data, signed by the attestation key and packaged with the certificate for that key.

Since this attestation key would almost certainly be planted in the hardware device before the platform was purchased by any buyer, one might claim that this achieves a certain level of anonymity. However, this mechanism provides for strong linkage between any two attestation events. If one of those events also reveals information about the owner of the platform, then this key ends up bound to that platform owner and constitutes an identifier of that owner. Therefore, we do not consider this mechanism to be anonymous.

Assuming compromise of a device's attestation key can be detected in the first place (cf., section 6), this mechanism gives the ability to revoke a compromised part, by publication of a list of compromised keys.

3.2 Shared Key Pair

One can generate a single key pair to be shared among all copies of a given device and certify that key to belong to one such device.

This achieves nearly perfect anonymity. An outside observer can not learn anything about the platform doing the attestation from the digital signature key

beyond that which is already learned in other ways (e.g., in the fact of the ability to do attestation, implying possession of the hardware device).

At the same time, revocation is not possible, short of revoking the entire class of devices or the entire process of attestation itself. Such a drastic consequence might drive some attacker to penetrate a copy of the hardware and publish its key, as an act of vandalism.

3.3 Blinding

There are well known methods for blinding a digital signature[3, 6]. With blinding, one introduces a third party that must be trusted by the challenger in an attestation protocol.

Using such a mechanism, a hardware device will have a permanent certified meta-attestation key. The device would generate as many additional attestation keys as it wishes, up to one per request. These can be generated as needed but for defeat of traffic analysis should be generated and certified ahead of time in batches.

This method requires a Trusted Third Party (TTP) whose purpose is to certify the newly generated anonymous keys. In the protocol between the hardware device and that TTP, the hardware device would generate a certificate body for the new attestation key, listing the TTP as issuer, hash that certificate body and send the blinded hash of the certificate body to the TTP for signing. That blinded hash would be in a message signed by the meta-attestation key and packaged along with the certificate on the meta-attestation key that had been generated by the device or platform manufacturer. The TTP would sign the blinded hash value and return it. The device would then unblind that signature and have a good signature on a certificate for the newly generated key, but without the TTP having any way to trace that newly generated key to the meta-attestation key (which can, in turn, be thought of as a permanent identifier of the device).

This method achieves anonymity but almost totally blocks revocation. One might detect that a given meta-attestation key has been used in a way impossible for a single hardware device. However, one can not revoke the meta-attestation key based on observed misbehavior of one of the attestation keys generated by that device and blindly certified by the TTP.

3.4 Anonymization

This method, like that of section 3.3, uses attestation keys generated by the device and those keys are sent to a TTP for certification. However, there is no blinding used. The TTP needs to be trusted by the user operating the prover not to keep records linking the user's platform's meta-attestation key to each allegedly anonymous attestation key.

If the TTP is truly trustworthy and keeps no such linkage and if the communication with that TTP is confidential, so that no eavesdropper can build such a linkage, then this achieves the same results as blinding, including the inability to revoke.

If the TTP does keep a linkage, then the TTP is in a position to revoke a meta-attestation key based on reported misbehavior of one of its "anonymous" attestation keys but this sacrifices anonymity.

3.5 Summary of Current Methods

We summarize these methods of attestation in Table 1:

method	**anonymity**	**revocation**
Individual Key	None	Good
Shared Key	Good	None
Blinding	Good	Poor
Anonymizer	Good if trustworthy	Good if not trustworthy

Table 1: Comparison of Attestation Methods

None of these methods achieves all of the goals stated in section 1. The Anonymizer method is the most tempting because it could satisfy either requirement, depending on conditions external to the design, however it can not satisfy both at once.

4 Attestation Pool

We propose doing attestation with a pool of shared keys. Under this method, a manufacturer of hardware devices requiring the ability to do attestation would initialize those devices with multiple keys. Each key would be present in multiple devices, giving anonymity via shared keys, as described in section 3.2. However, each device would be given a different set of keys. That set of keys would correspond to the device's identity.

Each of the keys would individually be certified as having been placed inside a valid hardware device and any challenger would have access to the certificates for each of the device's keys (perhaps because the device holds those certificates and provides them).

The protocol for doing attestation with a pool of keys would call for the challenger to provide not only freshness data (e.g., a nonce) with the challenge but also a list of all revoked attestation keys. The device processing this challenge, would select one unrevoked key from its list of keys and use that one key to attest to the requested data.

Since each device is identified by its full set of keys (and probably by any reasonable subset of its set of keys), if there is a chance for the challenger to

identify the device by some other means (e.g., IP address), the device should always use the same key in responding to that challenger. One way to do this, provided challengers are honest in their choice of revoked keys, is for each device to keep its keys in some fixed order and always to use the first unrevoked key on its list. Detecting dishonesty on the part of a challenger (e.g., one that keeps adding a device's keys to the revoked list until the challenger knows all of the device's keys and therefore its persistent identity) is left for future engineering design. One might, for example, have key revocation lists be signed by a single, unbiased source, independent of all challengers.

Since each device has a different set of keys, if one device were to be compromised and all of its keys published (as in the attack scenario described in section 3.2), all of the keys of that device could be revoked. The result would be revocation of that device alone. No other device would have all of those keys and therefore each other device would have at least one key that was not revoked.

This begs the question of how many keys can be revoked before some innocent device has no unrevoked keys to use for attestation. We address that question in section 5.

5 Selection of Pool Parameters

Assume that we will have M devices in active service, that each device will have a set of K keys taken from a total pool of N keys.

This produces a total of (MK) slots that can hold keys. If keys are distributed among devices uniformly, each key in the pool will appear in $\frac{MK}{N}$ different devices. However, if we assume also that each device shuffles its set of keys once, on initialization, and then always uses the first unrevoked key, each key will actually be used by only $\frac{M}{N}$ different devices, assuming no keys have been revoked. If there are revoked keys, each unrevoked key will be used by d devices, where

$$\frac{M}{N} \leq d \leq \frac{MK}{N} \tag{1}$$

with the interesting side effect that the more revoked keys there are, the "more anonymous" each unrevoked key becomes.

The US Census Bureau prohibits output data from permitting the discovery of any individual's data[9]. A similar rule here would imply that $d > 2$, however a manufacturer might well desire a much larger minimum value of d in order to make a stronger anonymity claim. If that minimum value of $d = d_0$, we get

$$d_0 \leq \frac{M}{N} \leq d \leq \frac{MK}{N} \tag{2}$$

$$d_0 N \leq M \tag{3}$$

5.1 Revocation

We assume that in practice a key will be revoked when it has been proved to have been compromised, however that is accomplished (cf., section 6). Keys might be revoked singly or in batches of up to K, depending on the care with which an attacker uses or reveals extracted keys.

For the purpose of analysis, let us assume that there are R revoked keys, chosen randomly from the set of N. We first compute two quantities:

$$T_1 = \binom{N}{K} \tag{4}$$

and

$$T_2(R) = \binom{R}{K} \tag{5}$$

where T_1 is the total possible number of different devices that could be populated with different sets of K keys taken from a total of N and $T_2(R)$ is the total number of different devices that could be populated entirely with R revoked keys.

Let us assume that when a device is initialized with keys, its set of keys was chosen at random (without replacement) from that set of T_1 possible sets. Let us also assume that keys are revoked at random, not because some device was compromised, but rather because the revocation algorithm is capricious. To a device that had been compromised, the revocation algorithm would not look random, but to an innocent device it would.

The probability of revocation of an innocent device would then be:

$$P_0(R) = \frac{T_2(R)}{T_1} \tag{6}$$

and we could expect $P_0(R)M$ innocently revoked devices. A manufacturer will want to choose parameters so that $P_0(R_{max})M \ll 1$, to minimize the probability of having a victimized innocent device owner for some maximum expected number of revoked keys, R_{max}.

By expansion of (6), we get

$$P_0(R) = \prod_{i=0}^{K-1} \frac{R-i}{N-i} < \left(\frac{R}{N}\right)^K \tag{7}$$

If $K \ll R_{max}$, we can approximate (7) with:

$$MP_0(R_{max}) \approx M\left(\frac{R_{max}}{N}\right)^K < E_V \tag{8}$$

where E_V is the tolerable expected number of innocent victims of revocation.

This, together with (3), gives the following inequalities to guide the choice of design parameters:

$$N \leq \frac{M}{d_0} \tag{9}$$

$$K < R_{max} < N \tag{10}$$

$$K > \frac{\ln(E_V) - \ln(M)}{ln(R_{max}) - ln(N)} \tag{11}$$

In the last relation, the numerator is the log of the tolerable probability of an innocent device revocation and the denominator is the log of the fraction of revoked keys.

6 Problem of Compromise Detection

The notion of revocation of attestation keys (or meta-attestation keys, as used in some methods) presumes that it is possible to detect the compromise of those keys. Assume, for example, that a hardware device has had its keys extracted by a physical attack[1, 2, 7, 5]. That attack on the hardware device could even be destructive. The keys obtained from the attack would be used in a software simulation of the hardware device.

If the challenger were able to inspect the computer and its hardware device physically, then a destructive attack might be detected. If the challenger is remote from the prover (hardware device), then physical inspection is not easy. That challenger might ask a human user at the computer in question to perform a physical inspection, but the quality of that inspection depends on the reliability (and skill) of the user involved.

Another method of detection depends on statistics of use of a given attestation or meta-attestation key. For example, if a key were to be released from the hardware that once protected it and were then sent to multiple parties around the Internet, it might be possible to detect use of one key from multiple places – a physical impossibility, if the key had been locked inside a single hardware component – or detect a key usage frequency that was physically impossible for a single computer and attestation device.

On the other hand, if someone were to extract a key and use it precisely normally, the remote challenger would not have any way to detect this key compromise. This might happen, for example, if an attacker were to gain momentary possession of a user's computer, penetrate the hardware device, extract its keys and then replace that device with an imposter device – one that would attest to falsehoods (as specified by the attacker) but use the exact same keys and protocols and timing that a legitimate device would use, when the imposter device

would be driven by the innocent user and therefore its usage profile would be completely innocent.

One method of preventing such attacks is to keep the hardware device (and the computer of which it is a part) under constant guard, so that no attacker has the opportunity to penetrate the hardware device.

Another method is to use very strong tamper resistance on the hardware device. Cryptographic processors used by banks often include motion sensors and other tamper detection mechanisms that drive the device to clear its storage of secrets between the time the attack is started and the time the attacker might have achieved penetration. One might also build devices that self-destruct physically in the event of suspected attack.

We note these problems merely because the subject of detection has been raised. We offer no simple solutions to this difficult engineering problem.

7 Conclusions

We have demonstrated in this paper a mechanism, using sets of private attestation keys none of which is unique to a given device, that achieves both anonymity and revocability. We have presented inequalities with which one can choose the parameters of such a system. To our knowledge, this is the only mechanism that achieves both of those attributes. It does not require trust in any third party other than the attestation device manufacturer.

This design has the weakness that an attacker intent on sabotaging the system by penetrating devices and publishing the keys thus discovered, forcing them to be revoked, can eventually cause a non-penetrated device to become non-functional because it would contain only revoked keys. Overcoming that weakness is an area for future research.

Finally, the ability to discover compromise of an attestation device appears to depend on the ability to have the local user prevent physical tampering. Tamper prevention is a separate area for engineering research, especially since few users can be expected to prevent hardware tampering by themselves.

References

1. Ross Anderson, Marcus Kuhn, "Low Cost Attacks on Tamper Resistant Devices", *Security Protocol Workshop*, April 1997, http://www.cl.cam.ac.uk/ftp/users/rja14/ramper2.ps.gz
2. Ross Anderson, "Reliability of Security Systems", http://www.cl.cam.ac.uk/users/rja14/#Reliability
3. D. Chaum, "Blind Signatures for Untraceable Payments", Advances in Cryptology Proceedings of Crypto 82, Plenum, pp. 199-203.
4. Comprehensive Test Ban Treaty, http://www.clw.org/pub/clw/coalition/ctbindex.htm

5. Paul Kocher, Joshua Jaffe, Benjamin Jun, “Differential Power Analysis”, Advances in Cryptology - CRYPTO ’99, Springer, 1999, pp. 388-397
6. Menezes, van Oorschot, Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1997, section 11.8.1.
7. Sergei Skorobogatov, Ross Anderson, “Optical Fault Induction Attacks”, http://www.cl.cam.ac.uk/ftp/users/rja14/faultpap3.pdf
8. Trusted Computing Platform Alliance, http://www.trustedpc.org/
9. U.S. Census confidentiality policy, http://www.census.gov/main/www/policies.html#confidential