Dr.Dobb's
THE WORLD OF SOFTWARE DEVELOPMENT

Enhanced Privacy ID

A remote anonymous attestation scheme for hardware devices

September 08, 2009
URL:http://www.drdobbs.com/security/enhanced-privacy-id/219501634

Ernie Brickell is a Senior Principal Engineer and Chief Security Architect at Intel. His e-mail is ernie.brickell@intel.com. Jiangtao Li is a Security Architect at Intel. His e-mail is jiangtao.li@intel.com.

Consider the following problem. A hardware device (for example, a mobile device, a graphics chip, a trusted platform module, a processor package, or a smart card) wants to prove to a verifier that it is a genuine hardware device manufactured by a certified hardware manufacturer. The easiest way to prove that it is the genuine article is for the verifier to read the serial number to the hardware manufacturer to verify that it is indeed the device in question. Each hardware device is assigned a unique serial number that is inscribed on the body of the device by the manufacturer. The problem with this solution is its limited application: the verifier needs to physically have the device in order for this kind of authentication to work. In many cases, a piece of hardware needs to be authenticated remotely. Remote hardware authentication is the main focus of this article.

A possible solution to the problem of remote hardware authentication is for the hardware manufacturer to assign each device a unique device certificate. More specifically, each device could be assigned a unique public and private key pair. The hardware manufacturer certifies the device by issuing a cryptographic certificate to the device public key. When the hardware device needs to be verified, it sends its device certificate to the verifier along with a signature signed with its private key. The verifier can then use the hardware manufacturer's public key to verify the device certificate and then use the device's public key in the certificate to verify the signature. This solution is secure, as long as the device can protect its private key, since only hardware devices made by the original manufacturer have valid device certificates.

This certificate approach is also scalable, as the device manufacturers can issue as many device certificates as they want. However, issuing a device certificate raises a privacy concern, because the device certificate is used to uniquely identify the device. The verifier, therefore, can use the device certificate to trace a device and the associated authentication activities.

In this article, we introduce a new cryptographic scheme called Enhanced Privacy ID (EPID) for remote, anonymous authentication of a hardware device. Using EPID, a hardware device can prove to a verifier remotely that it is a valid device, certified by the hardware manufacturer, without revealing its identity and without the verifier being able to link multiple authentication attempts made by the device.

Conceptually, an EPID scheme can be viewed as a special digital signature scheme. Unlike traditional digital signature schemes, one public key in the EPID scheme corresponds to multiple private keys. There are three types of entities in an EPID scheme: issuer, members, and verifiers. In our context, the issuer is the hardware manufacturer, the member is a hardware device made by the manufacturer, and the verifier could be software on the host, a server on the Internet, or another hardware device. The issuer creates an EPID public key and issues a unique EPID private key to each member. Each member can use this private key to digitally sign a message, and the resulting signature is called an EPID signature. The verifier can use the public key to verify the correctness of a signature, that is, to verify that the EPID signature was indeed created by a member in good standing with a valid private key. The EPID signature, however, does not reveal any information about which unique private key was used to create the signature.

Prerequisites and Design Requirements

We first formalize the remote hardware authentication problem, then describe the prerequisites for the problem, and end with our design requirements.

***Remote Hardware Authentication Problem**. To do remote authentication securely, cryptographic keys need to be used. There are three entities involved in a remote authentication scenario: the issuer, members, and verifiers. The issuer is a hardware manufacturer who creates a group. A member is a hardware device manufactured by the issuer. A member can join or leave the group.*

When a member joins the group as it is manufactured, the issuer issues a private key to the member. When the member leaves the group, the issuer revokes the private key of the member. Leaving the group is a rare event. It occurs only when the private key of the member (the hardware device) has been extracted from the hardware device, and the issuer has to revoke the membership of the device, or in other words, revoke the private key. A verifier is an entity that wants to know that a hardware device is a member of the group. The remote hardware authentication is an interaction between a member and a verifier. The member uses its private key to prove to the verifier that it is a valid member of the group and has not been revoked.

***Prerequisite**. To have a secure remote hardware authentication scheme, the member (that is, the hardware device) must have a good protection system for its private key. In other words, the member should have secure storage to store the private key and have a trusted execution environment to use the key to perform the membership proof. If an attacker can easily extract the key information from a member, then there is no way to do remote hardware authentication securely, as the attacker can always use the extracted private key to perform the membership proof before the key is revoked. This means that the verifier cannot tell whether the proof comes from the attacker or from a real hardware device.*

***Security Requirements**. The basic security requirement is straightforward; that is, only a member in good standing could perform the membership proof successfully. In other words, if the prover is not a member of the group, then its proof of membership would be rejected by the verifier. This property should hold unless a private key has been removed from a member and has not yet been revoked, or unless a problem considered computationally infeasible has been solved.*

***Privacy Requirements**. In a remote hardware authentication scheme, the membership proof must be anonymous and unlinkable. In addition, the private key of each member should be unknown to the issuer. More specifically, these are the required privacy properties:*

- *Given a membership proof, the verifier or the issuer cannot identify the actual prover, that is, cannot extract any identifiable information about the member from the proof. This is known as the anonymity property.*
- *Given two membership proofs, the verifier or the issuer cannot tell whether the proofs are generated by one member or by two different members. This is known as the unlinkability property.*
- *The issuer does not know any of the private keys of the members. Therefore, the issuer does not have a database of all the members' private keys.*

The unlinkability requirement is optional. In some applications, the verifier may require the membership proofs from a member to be linkable. Linkable proofs help to prevent a member from abusing the anonymity requirement. For example, suppose a verifier is a key-provisioning server that provisions a key to each member (that is, each hardware device). This verifier wants to make sure that each member is provisioned with only one key. Suppose that an adversary is able to extract a private key from a member device. If the remote hardware authentication scheme has the property of anonymity but not unlinkability, then the verifier would issue many keys to this adversary by using this one compromised member key. Then if the verifier found that one of the provisioned keys had been abused, he or she would be able to revoke it but would not be able to revoke all of the other keys that this adversary had obtained from the one compromised member key. Privacy issues with this use can be controlled, since the member needs to obtain the provisioned key from this verifier only once, and this usage can be unlinkable to any usage with any other verifier.

***Revocation Requirements**. A remote hardware authentication scheme must handle revocation. In general, when a hardware device is manufactured, it joins the group. Even if the ownership of the hardware device changes or the device is stolen, it is still a valid, authentic hardware device; thus, it is still in the group and does not need to be revoked.*

Only if the private key has been extracted from the secure storage of the hardware device, does the member have to be revoked. Given the prerequisites mentioned earlier, the issuer assumes that the member's (the hardware device's) private key is well protected. Thus, the revocation of a member is a rare event. However, the issuer needs to have the ability to revoke a member from the group if needed. The first revocation requirement is that the revocation of a group member should have minimum impact on the rest of the group members.

The second revocation requirement is that if an extracted private key is known to the issuer, then the issuer should be able to revoke that private key.

The third revocation requirement is that if a private key is used in a transaction, and it is later discovered that the key used in that transaction had been extracted, then this key should be revocable, even if it is not known. An example of such a case would be if a transaction was to provision a verifier key into a hardware device, and this verifier key was later shown to be extracted, the issuer may conclude that the private key of the hardware device has been corrupted and should be revoked.

Note that if an attacker extracts a private key from a hardware device and never uses the key, the key can probably never be revoked. On the other hand, if the attacker never uses the extracted private key to forge or emulate a hardware device, there is no damage to the hardware authentication scheme.

Application of Hardware Authentication

In this article, we present a new cryptographic scheme called Enhanced Privacy ID (EPID) that satisfies the security, privacy, and revocation requirements just discussed. Before we discuss EPID, however, we first discuss two applications of EPID. We first describe remote anonymous attestation and then discuss the application in digital drivers' licenses and identity cards.

***Remote Anonymous Attestation**. Why are we interested in the problem of remote hardware authentication? The answer lies in the fact that in many scenarios a verifier wants to know whether a request comes from an authentic hardware device or from a software emulator.*

Consider the following remote attestation example, depicted in Figure 1 as a conversation between a client and a service provider. A client platform is running a hardware-based trusted execution environment, based on a smartcard, or on a Trusted Platform Module (TPM). The trusted execution environment includes functionalities, such as secure code execution, secure data storage, and secure key generation. The platform requests a resource from a service provider, such as a Digital Rights Management (DRM) key. The service provider needs to determine whether the platform can protect its resource. The platform can do a remote attestation by sending the service provider a measurement of its computation environment, for example, the platform can send its hardware and software configuration. The attestation needs to be combined with a remote hardware authentication, that is, one signed by the hardware's private key. The logic of such an authentication is that an attacker can easily forge a measurement, but an attacker cannot compute a valid signature without knowing a valid hardware private key. A hardware authentication scheme satisfying the design requirements, outlined in the previous section, can provide both security and privacy for the remote attestation.

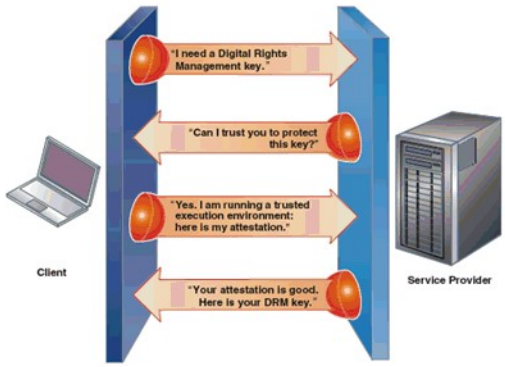

***Figure 1**: An Example of Remote Attestation (Source: Intel Corporation, 2009)*

The remote attestation problem was first introduced in the domain of a TPM, a small hardware module integrated into a platform, such as a laptop or a desktop. A direct anonymous attestation (DAA) scheme was developed by Brickell, Camenisch, and Chen [4] for remote authentication of a TPM, while preserving the privacy of the TPM. The DAA scheme was adopted by the Trusted Computing Group, an industry standardization body that aims to develop and promote an open industry standard for trusted computing hardware and software building blocks, and it was included in TPM specification version 1.2 [11].

Note that the EPID scheme presented here is an extension of the DAA scheme but has more revocation capabilities. Our EPID scheme has broader applicability beyond the remote attestation of a TPM. Let us look at two concrete applications of anonymous attestation.

***Secure E-Commerce and On-line Banking**. We now describe how EPID can be used for secure on-line banking. On-line banking is increasingly popular and provides great convenience to end users. However, the security of on-line banking is a concern, not only to end users but also to the banks. If the end user runs a platform that has a trusted execution environment and trusted I/O, the end user can conduct business in a relatively secure environment. However, the bank does not know whether the user is running in a secure environment. An anonymous attestation from the user's platform to the bank would give the bank more confidence that the transaction is secure.*

For example, if a bank user performs some high-volume transactions, the bank wants to make sure that the transactions are properly authorized. If the user runs a trusted execution environment, the user can use the EPID scheme to anonymously attest to the bank so that the bank can give a token to the platform for future transactions.

The bank would know that the token was being secured in a trusted execution environment. In later transactions, the user enters a password into the trusted execution environment that unlocks the token so that the bank can authenticate the user's environment. This assures the bank of the authenticity of the transaction.

***Content Protection**. Here we describe how EPID can be used to protect content. An on-line media server provides high-definition media content to clients. In order to download media content, each client needs to first download a Digital Rights Management (DRM) key from the server in order to decrypt the content. Before sending a DRM key to the client, the server wants to know whether the client can protect its DRM key. If the client has a hardware-based trusted execution environment and has a unique EPID private key embedded, it can use EPID to perform an anonymous attestation to the media server. After the attestation, the media server is convinced that the client is indeed running a trusted execution environment and is not in fact running a software emulator.*

Observe that in this example, if an attacker corrupts one EPID private key from a hardware device, the attacker may not publish the private key publicly. Instead, the attacker may use the compromised private key to obtain a DRM key from the media server. If the DRM key is found to be compromised on the Internet (for example, in ripper software), it can be traced back to the EPID private key that links it to the transaction that was used for provisioning the DRM key. Consequently, the issuer can revoke the compromised private key, based on the transaction of the key. This is an example in which the media server may wish to be assured that it issues only one DRM key for each EPID private key. This is accomplished through making the requests for DRM keys linkable to each other. But these requests would not be linkable to any other transactions.

***Drivers' Licenses and Identity Cards**. Various governments are considering including machine-readable information on drivers' licenses and identity cards. In the case of drivers' licenses, the machine-readable portion (for example, the bar code or magnetic strip) of the license is readable to anyone with a license reader. Unfortunately, such an approach raises serious privacy concerns, as personal information in the magnetic strip or bar code can be easily gathered and then sold without the owner's consent -- potentially leading to identity theft.*

Encrypting the machine-readable portion of the license has also been proposed. Such a practice poses significant key management challenges; the decryption must be available to authorized parties only.

We describe how EPID can be applied to drivers' licenses. Each license has an embedded smart card chip that can store and process information. A card reader is used to communicate with the license. The license is assigned a unique private key when it is issued by the government department that oversees the licensing of automobile drivers. It can be used for various purposes without violating the user's privacy. The smart card license would be able to prove to the reader that it is a valid license and that it has not been revoked, suspended, reported lost, and so forth. The smart card accomplishes this by using the proof of membership protocol; in this way the identity of the license is not revealed.

Each government agency would have multiple groups capable of issuing of licenses. During the process of proving its validity to a reader, a license reveals which license group it is in, and it reveals whether or not it is a valid license in good standing. It does not, however, reveal which license it is within that license group.

Using the EPID scheme, the membership proof is unlinkable. This means that if a license is used at a restaurant, for example, and it is valid and issued to someone of legal age, when that same license is used again at the same restaurant the next night, the restaurant owners would not be able to tell that it was the same license that was being presented. The restaurant owner is only acquainted with certain information: the validity of the license and that the patron is of legal age.

Overview of EPID

In our EPID scheme, there are three types of entities: issuer, members, and verifiers. There are two revocation lists: a list of corrupted private keys, denoted as PRIV-RL, and a list of signatures made from suspected extracted keys, denoted as SIG-RL. An EPID scheme has the following operations:

- *Setup. The issuer creates a public key and an issuing private key. The issuer publishes and distributes the public key to everyone (that is, to every member and every verifier).*
- *Join. This is an interactive protocol between an issuer and a member, the result of which is that the member obtains a unique private key.*
- *Sign. Given a message m and a SIG-RL, a member creates an EPID signature on m by using its private key.*
- *Verify. The verifier verifies the correctness of an EPID signature by using the public key. The verifier also checks that the key used to generate the signature has not been revoked in PRIV-RL or SIG-RL.*

Figure 2 depicts the interaction flows between the issuer, a member, and a verifier.

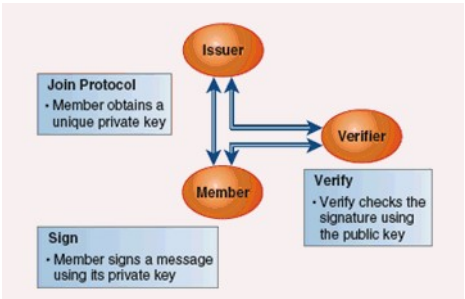

***Figure 2**: Basic EPID Scheme (Source: Intel Corporation, 2009)*

***Zero-knowledge Proofs**. In our EPID scheme, we use zero-knowledge proofs of knowledge [10] extensively. In a zero-knowledge proof system, a prover proves the knowledge of some secret information to a verifier such that (1) the verifier is convinced of the proof and yet (2) the proof does not leak any information about the secret to the verifier. In this article, we use the following notation for proof of knowledge of discrete logarithms. For example,*

$$\mathbf{PK}\ \{ (x) : y_1 = g_1^x \wedge y_2 = g_2^x \}$$

*denotes a proof of knowledge of integer **x** such that $y_1 = g_1^x$ and $y_2 = g_2^x$ hold, where **x** is known only to the prover, and g_1, y_1, g_2, y_2 are known to both the prover and verifier. In the above equation, **PK** stands for proof of knowledge and ∧ stands for logical conjunction.*

Proof of knowledge protocols can be turned into signature schemes by using the Fiat-Shamir heuristic [9]. In our EPID scheme, we develop several efficient zero-knowledge proof protocols for proving the knowledge of a valid EPID private key. In addition, we use an efficient zero-knowledge proof protocol developed by Camenisch and Shoup [8] for proving the inequality of discrete logarithms of two group elements y_1, y_2 to base z_1, and z_2, respectively, denoted as

$$\mathbf{PK}\ \{\ (x)\ :\ y_1 = z_1^{x} \wedge y_2 \neq z_2^{x}\}$$

Overview of Our Construction

*We begin with a high-level overview of our construction. In our scheme, each member chooses a unique membership key **f**. The issuer then issues a membership credential on f in a blind fashion such that the issuer does not acquire knowledge of the membership key **f**. The membership key and the membership credential together form the private key of the member. To sign a signature, the member proves in zero-knowledge that it has a membership credential on **f**. To verify a group signature, the verifier verifies the zero-knowledge proof.*

*In addition, each member chooses a base value **B** and computes $\mathbf{K} = \mathbf{B}^{\mathbf{f}}$. This (B, K) pair serves the purpose of a revocation check. We call B the base and K the pseudonym. To sign a signature, the member needs not only to prove that it has a valid membership credential, but also to prove that it constructs the (B, K) pair correctly, all in zero-knowledge.*

In EPID, there are two options to compute the base B: the random base option and the name base option.

- *Random base option. B is chosen randomly each time by the member. Under the decisional Diffie-Hellman assumption, no verifier can link two EPID signatures based on the (B, K) pairs in the signatures.*
- *Name base option. B is derived from the verifier's basename; for example, B = Hash (verifier's basename). Note that in this option, the value K becomes a pseudonym of the member with regard to the verifier's basename, as the member will always use the same K in the EPID signature to the verifier.*

*We first explain how membership can be revoked based on a compromised private key. Given a private key that has been revealed to the public, the issuer extracts the membership key **f** from the private key and inserts **f** into the private-key-based revocation list PRIV-RL. The issuer then distributes PRIV-RL to all the verifiers. Given an EPID signature, any verifier can check whether it was created with the corrupted private keys in PRIV-RL as follows:*

*Let (B, K) be the base-pseudonym pair in the EPID signature. The verifier can check that $\mathbf{K} \neq \mathbf{B}^{\mathbf{f}'}$ for every **f'** in PRIV-RL. If there exists an **f'** in PRIV-RL, such that $\mathbf{K} = \mathbf{B}^{\mathbf{f}'}$, it means that the signature was created with a revoked private key. Therefore the verifier can reject the signature.*

We now explain how membership is revoked, based on a transaction that a member was involved in. We call this kind of revocation signature-based revocation. Suppose a member's private key has been compromised by an attacker and has been used in some transaction. If the issuer has collected enough evidence to show that the private key used in the transaction was corrupted, the issuer can identify the EPID signature in the transaction and revoke the key, based on the signature. To do this, the issuer extracts the (B, K) pair from the signature and inserts the pair into the signature-based revocation list SIG-RL. The issuer then distributes the SIG-RL to all the verifiers. Before a member performs the membership proof, the verifier sends the latest SIG-RL to the member, so that the member can prove that it did not perform those transactions. More specifically, the member proves that it is not revoked in SIG-RL, by proving that, in zero-knowledge,

$$\mathbf{PK}\ \{(f)\ :\ K = B^{f} \wedge K' \neq B'^{f}\}$$

for each (B', K') pair in SIG-RL. If the zero-knowledge proof holds, the verifier is convinced that the member has not conducted those transactions and that membership has not been revoked.

Sketch of EPID Scheme

We have developed two EPID schemes, one from the strong RSA assumption [7] and the other from bilinear maps [6]. In this article, we briefly sketch the EPID scheme from bilinear maps. (The full scheme can be found in [6]).

*Let us first review some background on bilinear maps. Let **G1** and **G2** be two multiplicative cyclic groups of prime order p. Let g1 be a generator of **G1**, and $\mathbf{g_2}$ be a generator of **G2**. We say **e: G1 x G2 → GT** is an admissible bilinear map function, if it satisfies the following properties:*

*For all **u ∈ G1, v ∈ G2**, and for all integers **a, b**, equation $\mathbf{e(u^a, v^b) = e(u, v)^{ab}}$ holds. The result of $\mathbf{e(g_1 g_2)}$ is a generator of **GT**. There exists an efficient algorithm for computing **e(u, v)** for any **u ∈ G1, v ∈ G2**.*

Our EPID scheme is derived from Boneh, Boyen, and Shacham's group signatures scheme [2] and has the following operations:

***Setup**: The issuer does the following:*

1. *Chooses G1 and G2 of prime order p and a bilinear map function e : G1 x G2 → GT.*
2. *Chooses a group G3 of prime order p with generator g_3.*
3. *Chooses at random g_1 h_1, $h_2 \in G1$ and $g_2 \in G2$.*
4. *Chooses a random $r \in [1, p-1]$ and computes $w = g_2^{r}$. The public key is (g_1, g_2, g_3, h_1, h_2, w) and the issuing private key is r.*

***Join**: The join protocol is an interactive protocol between the issuer and a member as follows:*

1. *The member chooses at random f and y' from [0, p-1] and computes*

   $$T = h_1^{f} h_2^{y'}$$

2. *The member sends T to the issuer and performs the following proof of knowledge to the issuer: $PK\ \{(f, y')\ :\ T = h_1^{f}\ h_2^{y'}\}$*

   The issuer chooses at random x and y" from [0, p-1] and computes

   $$A = (g_1\ T\ h_2^{y''})^{\,1\,/\,(x\,+\,r)}$$

3. *The issuer sends (A, x, y") to the member.*
4. *The member computes y = y' + y"(mod p). The member's private key is (A, x, y, f).*

Note that given a valid private key (A, x, y, f), the following equation satisfies:

$$e(A,\ g_2^{x}\ w) = e(g_1 h_1^{f} h_2^{y},\ g_2)$$

***Sign**: Let (A, x, y, f) be the member's private key. The member does the following:*

1. *If the random base option is used, the member chooses B at random from G3.*
2. *If the name base option is used, the member computes B = Hash (verifier's basename).*

   Computes $K = B^f$

   Computes the following zero-knowledge proof

   $$PK\ \{(A, x, y, f) : e(A, g_2{}^x w) = e(g_1 h_1{}^f h_2{}^y, g_2) \wedge K = B^f\}$$

   This essentially proves that the member has a valid EPID private key issued by the issuer.

3. *Computes the following zero-knowledge proof*

   $$PK\ \{(f) : K = B^f \wedge K' \neq B'_f\}$$

   for each (B', K') pair in SIG-RL. This step proves that the member has not been revoked in SIR-RL; that is, the member did not create those (B', K') pairs in SIG-RL

4. *Converts all the above zero-knowledge proofs into a signature by using the Fiat-Shamir heuristic [9].*

***Verify**: Given the public key, PRIV-RL, SIG-RL, and an EPID signature, the verifier does the following:*

1. *If the random base option is used, the verifier verifies that B is an element in G3.*
2. *If the name base option is used, the verifier verifies that B = Hash (verifier's basename).*
3. *Verifies that K is an element in G3.*
4. *Verifies the following proof*

   $$PK\ \{(A, x, y, f) : e(A, g_2{}^x w) = e\ (g_1 h_1{}^f h_2{}^y, g_2) \wedge K = B^f\}$$

   This step verifies that the member has a valid EPID private key.

5. *Verifies that* $K \neq B^{f'}$ *for each f' in PRIV-RL. This step verifies that the member has not been revoked in PRIV-RL.*
6. *Verifies the following zero-knowledge proof*

   $$PK\{(f) : K = B^f \wedge K' \neq B'^f\}$$

   for each (B', K') pair in SIG-RL. This step verifies that the member has not been revoked in SIG-RL.

Comparison with Other Techniques

There are other techniques to remotely authenticate hardware, and in this section we review these techniques and compare them with our EPID scheme.

***Public Key Infrastructure (PKI)**. Each hardware device has a unique public and private key pair as well as a device certificate. To authenticate hardware by using PKI, the device simply shows its certificate to the verifier along with a signature created by using the device's private key. As mentioned previously, this PKI approach does not satisfy the privacy requirement.*

***Direct Anonymous Attestation (DAA)**. DAA was designed for anonymous attestation of TPM [4, 5]. DAA satisfies all the design requirements of remote hardware authentication; however, it has limited revocation capabilities compared to those of EPID. In the DAA scheme, there are two options for a balance between linkability and revocation. If the random base option is used, that is, a different base is used every time a DAA signature is performed, then any two signatures by a device are unlinkable, but revocation only works if the corrupted device private key has been revealed to the public. If a device has been compromised, but its private key has not been distributed to the verifiers (for example, if the corrupted device's private key is still under the control of the adversary), the corrupted TPM cannot be revoked. If the name base option is used, then any two signatures produced by a device, using the same base, are linkable. Thus, if the verifier determines that a device private key, used in a signature, has been compromised, that verifier can revoke that key locally; that is, the verifier can reject all future signatures generated by that private key, without knowledge of the compromised private key. However, the verifier cannot tell if a different verifier uses a different name base to revoke that private key, because when a different name is used, the revoked key cannot be identified. Furthermore, the name-based option does not safeguard privacy, because the verifier can link the transactions.*

***Group Signatures (GS)**. A group signature scheme [1, 2] has similar properties to those of the EPID scheme. In a group signature scheme, an issuer creates a group public key and issues unique private keys to each group member. Each group member can use the private key to sign a message, and the resulting signature is called a group signature. The verifier can verify a group signature by using the group public key. Unlike EPID, group signature schemes have an additional property called traceability. This property enables the issuer to open any group signature and identify the actual group member who created the signature. In other words, a group signature is anonymous to the verifiers but not to the issuer. Again, as compared to this scheme, EPID keeps the identity of the group member from the issuer.*

***Pseudonym System (PS)**. The pseudonym system [3], designed by Brands, can also be used for remote hardware authentication. In the pseudonym system, the display of a credential is anonymous by virtue of the fact that efficient zero-knowledge proof techniques are used for proving relations among committed values. To use the pseudonym system for hardware authentication, each hardware device obtains a credential from the issuer and uses the pseudonym credential for proof of membership. However, a credential in that system is linkable for multiple displays. To be unlinkable, a hardware device has to get multiple credentials from the issuer and use one credential at a time. This approach has limited application for hardware authentication, as the hardware device may never be able connect back to the issuer (the device manufacturer) once it has been produced. Thus, it cannot maintain the unlinkable property by continuing to get new credentials from the issuer.*

Summary

In Table 1, we summarize a comparison between different approaches to the remote hardware authentication problem. The EPID scheme is the only scheme that satisfies all the design requirements mentioned earlier.

Properties	PKI	DAA	Group Signatures	Pseudonym System	EPID
Unique Public Key	Yes	No	No	No	No
Unique Private Key	Yes	Yes	Yes	Yes	Yes
Anonymous	No	Yes	Yes	Yes	Yes
Unlinkable	No	Yes	Yes	No	Yes
Issuer Untraceable	No	Yes	No	Yes	Yes
Private-Key Revocation	Yes	Yes	Yes	Yes	Yes
Signature Revocation	Yes	No	No	Yes	Yes

***Table 1**: Approaches to Remote Hardware Authentication (Source: Intel Corporation, 2009)*

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. "A practical and provably secure coalition-resistant group signature scheme." In Advances in Cryptology -- Crypto, Volume 1880 of Lecture Notes in Computer Science, pages 255–270, 2000.

[2] D. Boneh, X. Boyen, and H. Shacham. "Short group signatures." In Advances in Cryptology -- Crypto, Volume 3152 of Lecture Notes in Computer Science, pages 41–55, 2004.

[3] S. Brands. Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge, MA, 2000.

[4] E. Brickell, J. Camenisch, and L. Chen. "Direct Anonymous Attestation." In Proceedings of the 11th ACM Conference on Computer and Communications Security, pages 132–145, 2004.

[5] E. Brickell, L. Chen, and J. Li. "A New Direct Anonymous Attestation Scheme from Bilinear Maps." In Proceedings of 1st International Conference on Trusted Computing, Volume 4968 of Lecture Notes in Computer Science, pages 166–178, 2008.

[6] E. Brickell and J. Li. "Enhanced Privacy ID from Bilinear Pairing." Cryptology ePrint Archive, Report 2009/095, 2009.

[7] E. Brickell and J. Li. "Enhanced Privacy ID: a Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities." In Proceedings of the 6th ACM Workshop on Privacy in the Electronic Society, pages 21–30, 2007.

[8] J. Camenisch and V. Shoup. "Practical Verifiable Encryption and Decryption of Discrete Logarithms." In Advances in Cryptology -- Crypto, Volume 2729 of Lecture Notes in Computer Science, pages 126–144, 2003.

[9] A. Fiat and A. Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In Advances in Cryptology -- Crypto, Volume 263 of Lecture Notes in Computer Science, pages 186–194, 1987.

[10] O. Goldreich, S. Micali, and A. Wigderson. "Proofs that Yield Nothing but their Validity." Journal of the ACM, Volume 38(3), pages 690-728, 1991.

[11] Trusted Computing Group. "TCG TPM Specification 1.2," 2003, http://www.trustedcomputinggroup.org

This article and more on similar subjects may be found in the Intel Technology Journal, June 2009 Edition, "Advances in Internet Security.