Capture Resilient ElGamal Signature Protocols*

Hüseyin Acan[1], Kamer Kaya[2,**], and Ali Aydın Selçuk[2]

[1] Bilkent University, Department of Mathematics
acan@fen.bilkent.edu.tr
[2] Bilkent University, Department of Computer Engineering
{kamer,selcuk}@cs.bilkent.edu.tr

Abstract. One of the fundamental problems of public key cryptography is protecting the private key. Private keys are too long to be remembered by the user, and storing them in the device which performs the private key operation is insecure as long as the device is subject to capture. In this paper, we propose server-assisted protocols for the ElGamal signature scheme which make the system capture resilient in the sense that the security of the system is not compromised even if the signature device is captured. The protocols also have a key disabling feature which allows a user to disable the device's private key in case both the device and the password of the user are compromised simultaneously.

1 Introduction

In public key cryptosystems, protecting the private key is a crucial problem. The keys are too long to be remembered by the users, and the devices which perform the private key operations may be subject to capture. This problem is even more significant if the device is a mobile authentication token such as a smart card or a hand-held device. An intuitive solution to this problem would be keeping the private key encrypted with the user's password in the device. However, in that case an offline dictionary attack on the password may be sufficient to obtain the private key if the device is captured, since the password by its very nature is a weak secret.

To make the devices resilient to capture, the private key can be stored in a server and the device can download it whenever a private-key operation is to be performed. To ensure that only the legitimate user can transfer the private key, an authentication to the server with the user's password is also required. However, the server itself has to be trusted and it must also be not subject to compromise in this solution.

MacKenzie and Reiter [4] avoided the requirement of a trusted server with a neat solution and gave a generic protocol framework to achieve capture resilience with different public key crpytosystems. They extended their technique

* This work is supported in part by the Turkish Scientific and Technological Research Agency (TÜBİTAK), grant number EEEAG-105E065.

** Supported by the Turkish Scientific and Technological Research Agency (TÜBİTAK) Ph.D. scholarship.

to develop RSA based signature [7] and ElGamal based decryption [2] protocols. The protocols also supported a key disabling feature if the device and the user's password are both compromised simultaneously.

What has been remarkably missing in the literature has been a capture resilient solution for ElGamal based signature systems [2]. ElGamal is the most commonly used signature algorithm after RSA, and it is especially significant for being the underlying algorithm of NIST's Digital Signature Standard [5]. Its elliptic curve versions are particularly popular in smart card authentication applications. In this paper, we extend the work of MacKenzie and Reiter to the ElGamal signature protocols and show how capture resilience can be achieved in the systems which use this algorithm for authentication. We consider two different versions of the ElGamal signature algorithm and present a capture resilient protocol for each of them.

The rest of this paper is organized as follows: In Section 2, we summarize the related background. After explaining the notation and the framework in Section 3, we present the capture resilient ElGamal signature protocols in Section 4. Section 5 concludes the paper with ideas for future research.

2 Background

The problem of using public key authentication techniques where the users are able to remember only weak passwords has been studied in the literature for several years. Capture resilience is an important feature of such techniques since the devices are subject to being captured. Obviously, if the private key of a device is stored in the device, the system would not be capture resilient. In 1999, Perlman and Kaufman [6] designed a strong password protocol where the private key is not stored in the device and downloaded from the server when needed. This idea requires an additional authentication mechanism which uses only a weak password to ensure that the private key is downloaded only by the legitimate user.

The idea of storing the private key in the server can also be used for obtaining capture resilience. However, storing the private key of the device in the server is secure only if the server is trusted and not subject to compromise. It is obvious that by capturing the server, an adversary can mount an offline dictionary attack even if the private key is encrypted with the weak password. MacKenzie and Reiter [4] solved this problem by storing a ticket in the device which includes the private key information. This ticket is encrypted with the public key of the server and contains the private key information encrypted with the password. Based on this idea, they proposed a generic private key retrieval protocol which does not require the server to be trusted.

MacKenzie and Reiter extended their ideas in the generic private key retrieval protocol and proposed RSA-based signature and ElGamal-based decryption protocols which supported capture resilience and also key disabling. Key disabling preserves the security of the system in case that the device is captured and the user password is compromised. In that case, the legitimate user can disable his

private key by sending a request to the server and, if this request is valid, the server will ignore future signature/decryption requests with tickets associated with the old private key. To achieve capture resilience, MacKenzie and Reiter proposed to divide the private key information into two. The first one, the server's share, is encrypted with the public key of the server and stored in the device whereas the second one is produced from the password and a random number stored in the device. The protocols proposed in [4] are secure if one of the device or the server is compromised alone; however, if both are compromised simultaneously, the password can be broken by a dictionary attack and the private key can be obtained.

3 Framework

In our framework, the device and the server are connected through a public network where the attackers are able to eavesdrop and inject messages. Both the device and the server has a public-private key pair, (PK_{dvc}, SK_{dvc}), and (PK_{svr}, SK_{svr}), respectively. We use E and D to denote the public key encryption and decryption operations with these keys. The legitimate user of the device creates a valid ElGamal signature for a message m by cooperation with the server. The cooperation is needed since the device cannot use SK_{dev} without the help from the server. The security of the signing operation is provided by a user password denoted by π.

We follow the attack and adversary models of MacKenzie and Reiter [4] where there are various adversary types. Each adversary type is assumed to be able to control the network, so an adversary can read all the messages going through the network and control the inputs to the device and the server. In addition, an adversary can capture a subset of $\{\pi, dvc, svr\}$ and learn all of the static contents of the captured resources. The type of an adversary is specified with respect to a capture set $\mathcal{S} \subset \{\pi, dvc, svr\}$ where $ADV(\mathcal{S})$ denotes an adversary who captured the components in $\mathcal{S}$.

Here we give the definition and notation for the ElGamal signature schemes used in the proposed protocol:

3.1 ElGamal Signatures

Let g be a generator of $\mathbb{Z}_p^*$ where p is a large prime. The private key is $SK_{dvc} = x$ where $\gcd(x, p-1) = 1$. The public key is $PK_{dvc} = (y, g, p)$ where $y = g^x \bmod p$. To sign a message m, the signer chooses a random ephemeral key k such that $\gcd(k, p-1) = 1$ and computes $r = g^k \bmod p$. Then he computes the signature s by using a cryptographic hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and sends $\sigma = (r, s)$ as the signature. The receiver can verify the signature by checking an equation in modulo p. A detailed discussion of the ElGamal signature scheme can be found in [2].

There are several variants of the ElGamal signature algorithm where s is defined differently [3]. Two important variants used in our protocols are as follows:

1. In the first variant, the signature is computed as

$$s = xr + kh(m) \bmod (p-1)$$

and the verification equation is

$$g^s \stackrel{?}{=} y^r r^{h(m)} \bmod p.$$

2. In the second one, the signature is computed as

$$s = xh(m) + kr \bmod (p-1)$$

and the verification equation is

$$g^s \stackrel{?}{=} y^{h(m)} r^r \bmod p.$$

4 Capture Resilient ElGamal Signature Protocols

In this section, we propose two capture resilient signature protocols for the ElGamal signature algorithm. The protocols proposed here are based on the ideas in [4] but differ significantly in the details since the distribution techniques of MacKenzie and Reiter do not apply due to the significantly different nature of the ElGamal signature algorithm. In our protocol, the private key of the user is shared between the server and the device. When a user wants to sign a message m, the device, which was previously initialized with π by the legitimate user, signs m by cooperation with the server. At the first step of our protocol, the device is authenticated to the server with π and sends the server's share of the private key x with m. One part of the signature is computed by the server with its share of x. Upon receiving the partial signature from the server, the device computes the other part with its share which is produced from π.

As mentioned above, proposed protocols also support a key disabling operation when both π and *dvc* are compromised. Disabling a key means sending a message to the server to ignore the future requests with the corresponding private key. Hence, the adversary cannot forge signatures by cooperation with the server after key disabling.

We present the details of the protocol in two phases. The first phase is the device initialization phase where the required data for the protocols are generated in the device. Some of these data are stored in the device and some are deleted immediately. The second phase is the signature protocol where the device signs a message with the help of the server. After introducing the signature protocols, we describe the key disabling operation and then we conclude this section with the security analysis.

4.1 Device Initialization

The static data required for the protocols is produced in the device initialization phase. The notation $A \leftarrow_R B$ denotes that A is randomly chosen from the elements of B with a uniform distribution. In the proposed protocols, random numbers are chosen from κ-bit numbers where κ can be considered as a security parameter for the protocols:

$$
\begin{aligned}
t &\leftarrow_R \{0,1\}^\kappa \\
u &\leftarrow_R h_{dsbl}(t) \\
v &\leftarrow_R \{0,1\}^\kappa \\
a &\leftarrow_R \{0,1\}^\kappa \\
b &\leftarrow h(\pi) \\
x_1 &\leftarrow f(v,\pi) \\
x_2 &\leftarrow x - x_1 \bmod (p-1) \\
\tau &\leftarrow E_{PK_{svr}}(a,b,u,x_2,g,p)
\end{aligned}
$$

The values t and u are computed to be used in the key disabling phase. The random number a is chosen to be used in generating the message authentication codes(MAC). The function f takes two arguments and computes $x_1 \in \mathbb{Z}_p$, the device's share of x. The ticket τ, which contains x_2, the server's share of x, is encrypted with PK_{svr} to guarantee that an adversary cannot see the share. The values v, a, t and τ are stored on the device and, u, b, x, x_1, x_2 and π are deleted from the device immediately after the device initialization.

4.2 Signature Protocol

The device starts this phase with the values stored in the device initialization phase, the password π and the message m to be signed by using the private key x of the device. The two variants of our protocols are illustrated in Fig.1 and Fig.2. These protocols differ only in the signature equation, hence the notation used in both is same. To start the protocol the device first chooses two κ-bit random numbers ρ and k_1 and then computes β, the hash of π. Then $r_1 = g^{k_1} \bmod p$ is computed to generate r used in the ElGamal signature scheme described in the previous section. The protocol message γ is obtained by encrypting m, β, ρ and r_1 with PK_{svr} and sent to the server with the ticket τ and a MAC $\delta = mac_a(\gamma, \tau)$.

Upon receiving γ, δ and τ, server first checks if the messages and the password π used by the device are authentic. Then it chooses a random number k_2 and computes $r_2 = g^{k_2} \bmod p$. Note that k_1 and k_2 are random numbers whose sum, k, is the ephemeral key used in the ElGamal signature scheme. After computing r, the server computes the partial signature s_2 and sends them to the device by masking them with ρ.

DEVICE		SERVER
$\beta \leftarrow h(\pi)$		
$\rho \leftarrow_R \{0,1\}^\kappa$		
$k_1 \leftarrow_R \{0,1\}^\kappa$		
$r_1 \leftarrow g^{k_1} \bmod p$		
$\gamma \leftarrow E_{PK_{svr}}(m, \beta, \rho, r_1)$		
$\delta \leftarrow mac_a(\gamma, \tau)$		
	$\xrightarrow{\gamma,\delta,\tau}$	
		$(a, b, u, x_2, g, p) \leftarrow D_{SK_{svr}}(\tau)$
		abort if $mac_a(\gamma, \tau) \neq \delta$
		$(m, \beta, \rho, r_1) \leftarrow D_{SK_{svr}}(\gamma)$
		abort if $\beta \neq b$
		$k_2 \leftarrow_R \{0,1\}^\kappa$
		$r_2 \leftarrow g^{k_2} \bmod p$
		$r \leftarrow r_1 r_2 \bmod p$
		$s_2 \leftarrow (x_2 r + k_2 h(m)) \bmod (p-1)$
		$\eta \leftarrow \rho \oplus s_2$
	$\xleftarrow{\eta,r}$	
$s_2 \leftarrow \rho \oplus \eta$		
$x_1 \leftarrow f(v, \pi)$		
$s_1 \leftarrow (x_1 r + k_1 h(m)) \bmod (p-1)$		
$s \leftarrow (s_1 + s_2) \bmod (p-1)$		
abort if $g^s \neq y^r r^{h(m)}$		
return (r, s)		

Fig. 1. Capture resilient ElGamal signature protocol, the 1^{st} variant: $s = xr + kh(m)$

DEVICE		SERVER
$\beta \leftarrow h(\pi)$		
$\rho \leftarrow_R \{0,1\}^\kappa$		
$k_1 \leftarrow_R \{0,1\}^\kappa$		
$r_1 \leftarrow g^{k_1} \bmod p$		
$\gamma \leftarrow E_{PK_{svr}}(m, \beta, \rho, r_1)$		
$\delta \leftarrow mac_a(\gamma, \tau)$		
	$\xrightarrow{\gamma,\delta,\tau}$	
		$(a, b, u, x_2, g, p) \leftarrow D_{SK_{svr}}(\tau)$
		abort if $mac_a(\gamma, \tau) \neq \delta$
		$(m, \beta, \rho, r_1) \leftarrow D_{SK_{svr}}(\gamma)$
		abort if $\beta \neq b$
		$k_2 \leftarrow_R \{0,1\}^\kappa$
		$r_2 \leftarrow g^{k_2} \bmod p$
		$r \leftarrow r_1 r_2 \bmod p$
		$s_2 \leftarrow (x_2 h(m) + k_2 r) \bmod (p-1)$
		$\eta \leftarrow \rho \oplus s_2$
	$\xleftarrow{\eta,r}$	
$s_2 \leftarrow \rho \oplus \eta$		
$x_1 \leftarrow f(v, \pi)$		
$s_1 \leftarrow (x_1 h(m) + k_1 r) \bmod (p-1)$		
$s \leftarrow (s_1 + s_2) \bmod (p-1)$		
abort if $g^s \neq y^{h(m)} r^r$		
return (r, s)		

Fig. 2. Capture resilient ElGamal signature protocol, the 2^{nd} variant: $s = xh(m) + kr$

When the device gets r and $\eta = s_2 \oplus \rho$, it computes his partial signature and combines it with the one sent by the server. Then it checks the validity of s and outputs the signature if it is valid.

4.3 Key Disabling

It is obvious that an adversary $ADV\{dvc, \pi\}$ can forge signatures by impersonating the user of the device. In the proposed protocols, the legitimate user has a key disabling option to disable his private key in these cases. To start the key disabling mechanism, the legitimate user sends τ and t to the server, hence these values must be kept offline. Upon receiving a disabling request, the server decrypts the ticket and checks if $u \stackrel{?}{=} h_{dsbl}(t)$ where h_{dsbl} is a hash function. If the equality holds, server puts τ to the disabled list and refuses to respond to requests with this ticket.

4.4 Security Analysis

As stated earlier, the device is connected to a server through a public network and adversaries are able to see and control the messages to the device and the server. Here we discuss the security of the proposed protocols against four types of adversaries:

1. $ADV\{dvc\}$: This adversary captured the device, but cannot initiate the protocol since he does not know π. However, he can perform an online guessing attack to find π by observing the responses of the server to his invocations. The success probability of this attack is simply $q/|D|$ where D is the password space and q is the number of invocations. Given that only a negligible fraction of the password space can be tried in an online guessing attack, the system is secure against this type of attack.

   The only information the adversary can obtain by eavesdropping is the signature of a message m since all the messages are encrypted. He can obtain neither the partial signatures nor any other information related to them.

2. $ADV\{\pi, svr\}$: This adversary type has all the static data stored in the server and has also obtained the password. He knows the public values PK_{dvc}, g, p and PK_{svr}. Furthermore, he knows SK_{svr}; so he can decrypt the ticket τ and find the values a, b, u and x_2. He can also obtain r_1, ρ, and m by decrypting γ.

   For this type of adversary, obtaining the private key x is equivalent to obtaining x_1. Having π is not sufficient for an offline dictionary attack since v is also required, which is unknown to the adversary. By eavesdropping or impersonating the server, he can obtain valid s_2 values for a valid signature s. The partial signature of the device, s_1, can be computed by subtracting s_2 from s. However, s_1 is not sufficient to obtain x_1 since k_1 is also required

but it is an unknown for the adversary. Note that k_1 cannot be obtained from r_1 because this is equivalent to the discrete log problem.

3. $ADV\{dvc, svr\}$: If all components of the system are broken but the password π, the attacker can always try to find π by an offline dictionary attack. In this case, the dictionary attack will be done by comparing the hash of the password guess to b, which can be obtained by decrypting the ticket. If the password is broken, the attacker can forge signatures.

   If the password is not broken, obtaining the private key x will be equivalent to obtaining x_1 for this type of adversary. By eavesdropping or impersonating the server, he can obtain a valid partial signature of the server, s_2, for a valid signature s so he can also obtain s_1. However, these values are not sufficient to obtain x_1 because similar to the $ADV\{\pi, svr\}$ this adversary type suffers from the lack of k_1 which is an unknown to the adversary.

4. $ADV\{\pi, dvc\}$: If the device and the password are compromised simultaneously, the attacker will be virtually indistinguishable from the legitimate user and can forge signatures by cooperation with the server until the legitimate user of the device disables his private key. For this type of adversary, obtaining the private key is equivalent to obtaining x_2. With a procedure similar to the ones above, this adversary can obtain s_1 and s_2 for a valid signature s however he still suffers from the lack of k_2 to obtain x_2.

5 Conclusions and Future Work

In this paper, we proposed two ElGamal based signature protocols for capture resilient devices. It is obvious that storing the private keys in the device make the devices vulnerable to capture. Encrypting the private key with a weak password is also insufficient since an offline dictionary attack can always be performed once the device is captured. In this paper, we extended the approach of MacKenzie and Reiter [4] to the ElGamal signature scheme which is the fundamental of many important systems. The proposed protocols provide a server assisted solution for ElGamal-based authentication which makes the systems resilient to capture of the devices.

Future work on this problem may include a formal security analysis of the proposed protocols possibly using the random oracle model [1]. Techniques used by MacKenzie and Reiter [4] to prove the security of their protocols for RSA signature and ElGamal decryption protocols do not directly apply to the ElGamal signature scheme due to the more involved structure of the latter. A formal proof of a similar nature may be possible but would require more creative approaches with the random oracle model.

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[2] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[3] P. Horster, H. Petersen, and M. Michels. Meta-elgamal signature schemes. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 96–107, New York, NY, USA, 1994. ACM Press.

[4] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. *International Journal of Information Security*, 2(1):1–20, 2003.

[5] NIST. The digital signature standard. *Commun. ACM*, 35(7):36–40, 1992.

[6] R. Perlman and C. Kaufman. Secure password-based protocol for downloading a private key. In *Proc. of Network and Distributed System Security*, 1999.

[7] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.