A Computational Diffie–Hellman based Server Supported Signature

Augustin P. Sarr, Togdé Ngarenon

Abstract— The storage of private keys on mobile devices is a real issue; hardware based solutions are inconvenient and password encrypted storages may be vulnerable to offline dictionary attacks. At ESORICS 2017, Buldas *et al.* proposed an efficient (software only) password based server supported signature scheme, geared to mobile devices, termed Smart–ID. Unfortunately, despite its numerous interesting features, Smart–ID has two main limitations: (*i*) the (four prime RSA) signatures are large, and (*ii*) the design offers no password update mechanism. We propose an efficient password based server supported signature scheme, termed SCM, which achieves all the security attributes of Smart–ID, in addition to offering a password update mechanism and yielding smaller signatures on ECC groups. Under the computational Diffie–Hellman and the random oracle model, SCM is unforgeable against chosen message attacks.

Index Terms— Computational Diffie–Hellman, Shared Chevallier–Mames (SCM) Signature, Server–Supported Signature Scheme, Clone Detection, Password Update, Smart–ID.

1 Introduction

DIGITAL signatures are widely used in every day communications, either directly or as building blocks for more complex cryptographic schemes. Mobile devices are largely involved in these communications. On these devices, the issue of a secure storage of the private keys cannot be conveniently handled with a classical hardware module, which stores the private keys. Connecting a hardware (a smart–card reader, for instance) to these devices does not seem to be convenient. On another side, a software only solution which consists in encrypting the private keys with a user password may yield reference points for offline dictionary attacks.

In the case of RSA signatures, Buldas *et al.* [4] propose an interesting solution, termed Smart–ID, built upon Damgård *et al.*'s four prime RSA [6]. Smart–ID is improved in [9]. Unfortunately, the Smart–ID design has two significant limitations. The signatures are large, for instance, for a target of 128–bits of security, it yields 6144–bits signatures; and, it offers no password update mechanism to device owners.

In this paper, we explore the design of a password based server supported signature scheme, geared to software only implementations on mobile devices. The aim is to propose a scheme which offers all the security attributes achieved in Smart–ID, together with a password update mechanism, in addition to yielding smaller signatures.

Basing on Chevallier–Mames' signature scheme, we propose an efficient design, termed SCM, which is unforgeable against chosen message attacks, under the computational Diffie–Hellman problem and the Random Oracle (RO) model. Similar to Smart–ID, it is secure against a malicious client (*i. e.* a client cannot generate a signature without the help of the server), against a malicious server, against device read, and integrates a clone detection mechanism. And, contrary to Smart–ID, (*i*) it offers a password update mechanism, and (*ii*) it can be efficiently instantiated on the group of rational points of an elliptic curve, wherein it yields, for a target of 128 bits of security, for instance, signatures that are six times smaller than the Smart–ID ones.

This paper is organized as follows. In Section 2, we recall some preliminaries. In Section 3, we present the SCM scheme. We provide a security and efficiency comparison between SCM, Smart–ID and a smart–card in section 4, and some concluding remarks in Section 5.

- *Augustin P. Sarr received his PhD in Cryptography at the University of Grenoble (France). He is an Assistant Professor at the Department of Applied Mathematics of the University Gaston Berger (Senegal). Email: augustin-pathe.sarr@ugb.edu.sn*
- *Togdé Ngarenon is currently pursuing PhD degree program in Cryptography af the University Gaston Berger. E-mail: tognice2015@gmail.com*

2 Background

2.1 Notations

$\mathcal{G} = \langle G \rangle$ is a cyclic group of prime order p with identity element $1_{\mathcal{G}}$. We denote by $Exp(\mathcal{G}, t)$ the computational cost of t exponentiations with $|p|$–bits exponents in $\mathcal{G}$; $Exp(\mathcal{G})$ is a shorthand for $Exp(\mathcal{G}, 1)$. For an integer n, $[n]$ denotes the set $\{0, \cdots, n\}$. If $\mathcal{S}$ is a set, $a \leftarrow_{\mathrm{R}} \mathcal{S}$ means that a is chosen uniformly at random from $\mathcal{S}$. For a probabilistic algorithm $\mathcal{A}$ with parameters $u_1, \cdots, u_n$ and output $V \in \mathbf{V}$, we write $V \leftarrow_{\mathrm{R}} \mathcal{A}(u_1, \cdots, u_n)$. If $x_1, x_2, \cdots, x_k$ are objects belonging to different structures, $(x_1, x_2, \cdots, x_k)$ denotes a representation as a bit string of the tuple such that each element can be unequivocally parsed.

2.2 Computational Diffie–Hellman (CDH) Assumption

If $X, Y \in \mathcal{G}$, we denote the integer $x \in [p-1]$ such that $G^x = X$ by $\log_G X$, and $Y^{\log_G X}$ by $CDH(X, Y)$. The CDH assumption is said to hold in $\mathcal{G}$ if for all efficient algorithm $\mathcal{A}$,

$$Adv_{\mathcal{A}}^{CDH}(\mathcal{G}) = \Pr\begin{bmatrix} X \leftarrow_{\mathrm{R}} \mathcal{G}; Y \leftarrow_{\mathrm{R}} \mathcal{G}; \\ Z \leftarrow_{\mathrm{R}} \mathcal{A}(G, X, Y) : Z = CDH(X, Y) \end{bmatrix}$$

is negligible in k.

2.3 Digital Signature

A Digital Signature is a triple of efficient algorithms $\mathcal{S} = (Gen, Sign, Vrfy)$ together with a message space $\mathbf{M}$, such that:

- Gen is probabilistic algorithm which takes a domain parameter dp as input and outputs a key pair (sk, pk); sk is the signing key and pk is public.
- $Sign$ takes as inputs sk and a message $m \in \mathbf{M}$ and outputs a signature σ.
- $Vrfy$ is deterministic; it takes as inputs pk, a message m, and a signature σ and outputs $d \in \{0,1\}$.
- And $\mathcal{S}$ is such that for all $((sk, pk), m) \in \{Gen(dp)\} \times \mathbf{M}$,

$$\Pr[Vrfy(pk, m, Sign(sk, m)) = 1] = 1.$$

1) $dp \leftarrow Setup(1^k)$; $(sk, pk) \leftarrow_R Gen(dp)$;
2) $(m_0, \sigma_0) \leftarrow_R \mathcal{A}^{\mathcal{O}_{Sign}(\cdot)}(pk)$, where $\mathcal{O}_{Sign}(\cdot)$ is a signing oracle which takes as input a message m and outputs $\sigma \leftarrow_R Sign(sk, m)$.
3) The attacker $\mathcal{A}$ succeeds if : (*a*) $Vrfy(pk, m_0, \sigma_0) = 1$ and (*b*) $m_0 \notin M_{\mathcal{A}}$, where $M_{\mathcal{A}}$ is the set of messages issued to the signing oracle.

We denote by $Adv_{\mathcal{A},\mathcal{S}}^{uf-cma}(1^k)$ the probability that $\mathcal{A}$ wins the game.

Game 1 UF-CMA Security Game

Definition 1 *A signature scheme $\mathcal{S}$ is said to be (t, q_s, ε) unforgeable against chosen message attacks (UF-CMA) if for all adversaries $\mathcal{A}$ playing Game , running in time t, and issuing q_s signature queries, $Adv_{\mathcal{A},\mathcal{S}}^{uf-cma}(1^k) \leqslant \varepsilon$.*

2.4 The Random Oracle Model

For a given security parameter k, we assume that the hash functions are Random Oracles [1]. *i. e.* if $H: \{0,1\}^* \leftrightarrow \{0,1\}^{2k}$ is a hash function, we assume that:

a) for all $x \in \{0,1\}^*$, $H(x) \in \{0,1\}^{2k}$, and
b) for all $m \in \mathbb{N}$, for all $x_1, x_1, \cdots, x_m \in \{0,1\}^*$, for all $y_1, y_2, \cdots, y_m \in \{0,1\}^{2k}$, for all $x_0 \notin \{x_1, \cdots, x_m\}$, and for all $y_0 \in \{0,1\}^{2k}$,
$\Pr(H(x_0) = y_0 | H(x_1) = y_1 \wedge \cdots \wedge H(x_m) = y_m) = 2^{-2k}$.

3 The SCM Signature Scheme

3.1 Chevallier–Mâmes' (CM) Signature

We recall in Figure 1 the CM signature scheme; it is tightly unforgeable against chosen message attacks under the CDH assumption and the RO model [5, Theorem 2]. We do not mention group membership tests which are negligible in $\mathbb{Z}_q^*$ and elliptic curve groups; $H_1: \{0,1\}^* \to \mathcal{G}$, and $H_2: \{0,1\}^* \to [p-1]$ are hash functions.

Remark 1

a) *The group element X at step 1 of the $Sign_{CM}$ algorithm is termed ephemeral key. If CM is modified such that instead of computing $R \leftarrow H_1(X)$ and $h \leftarrow H_2(m, G, X, R, V, W, pk)$ in the $Sign_{CM}$ and $Vrfy_{CM}$ algorithms, one computes $R \leftarrow H_1(X, aux)$ and $h \leftarrow H_2(m, G, X, R, V, W, A \cdot pk)$, for some fixed bit string* aux *and some fixed $A \in \mathcal{G}$, then the Theorem 2 of [5] still holds for the resulting CM variant. Given $aux \in \{0,1\}^*$ and $A \in \mathcal{G}$, we denote the signature generation and verification algorithms in the CM variant using* aux *and A by $Sign_{VCM(aux,A)}(\cdot,\cdot)$ and $Vrfy_{VCM(aux,A)}(\cdot,\cdot,\cdot)$, respectively.*

$Gen_{CM}(dp)$: $sk \leftarrow_R [p-1]$; $pk \leftarrow G^{sk}$; return (sk, pk);
$Sign_{CM}(sk, m)$:
1) $x \leftarrow_R [p-1]$; $X \leftarrow G^x$; $R \leftarrow H_1(X)$; $V \leftarrow R^x$;
2) $W \leftarrow R^{sk}$; $h \leftarrow H_2(m, G, X, R, V, W, pk)$;
3) $\sigma \leftarrow x + h \cdot sk \bmod p$; return (W, σ, h);
$Vrfy_{CM}(pk, m, (W, \sigma, h))$:
1) $X \leftarrow G^{\sigma} pk^{-h}$; $R \leftarrow H_1(X)$; $V \leftarrow R^{\sigma} W^{-h}$;
2) If $h = H_2(m, G, X, R, V, W, pk)$ then return 1;
3) else return 0;

Figure 1 Chevallier–Mâmes' Signature Scheme

3.2 Description of SCM

We assume a secure channel between the client Clt and server Srv; such a channel may be obtained using a password based key exchange [2, chap.8], for instance.

3.2.1 Setup

The algorithm is the same as for CM ; it takes a security parameter k as an input and defines a group $\mathcal{G} = \langle G \rangle$ of prime order p, together with the digest functions $H_1: \{0,1\}^* \to \mathcal{G}$, and $H_2: \{0,1\}^* \to [p-1]$. It outputs $dp = (\mathcal{G}, G, p, H_1, H_2)$.

3.2.2 Shared Key Pair Generation

The aim is to generate a public key pk together with a corresponding private key sk such that: (*i*) pk is uniformly distributed over $\mathcal{G}$, (*ii*) sk is additively shared between the client Clt and the server Srv, (*iii*) Clt's private share a_1 and Srv's share a_2 are independent, and (*iv*) Clt or Srv alone cannot generate a signature. The key generation is as in Protocol 1; the execution aborts if any verification fails.

A client's private share is uniformly distributed over $[p-1]$

a) The client Clt, with password $Pwd \in \mathcal{P}$ does the following: $r, u \leftarrow_R [p-1]$; $a_1 \leftarrow H_2(u, Pwd)$; $A_1 \leftarrow G^{a_1}$; $h_a \leftarrow H_2(r, A_1)$; send h_a to the server;
b) At receipt of h_a, Srv does the following: $a_2 \leftarrow_R [p-1]$; $A_2 \leftarrow G^{a_2}$; $y \leftarrow_R [p-1]$; $Y \leftarrow G^y$; send (h_a, A_2, Y) to Clt;
c) At receipt of (h_a, A_2, Y), Clt does the following:
   1) Send (r, A_1) to Srv;
   2) Compute $pk \leftarrow A_1 A_2$;
   3) Store (u, r, Y, pk), and delete the other values.
d) At receipt of (r, A_1), Srv does the following:
   1) Verify that $h_a = H_2(r, A_1)$;
   2) $pk \leftarrow A_1 A_2$; $T_S \leftarrow T_0$;
   3) Store (a_2, r, y, Y, pk, T_S) and delete the other values.
e) The public key is pk

Protocol 1 SCM Key Pair Generation

and independent from the server's share. Besides, when Srv

generates A_2, it is only aware of h_a, under the RO model, it cannot then infer any information about A_1. Following [9], in addition to the shared key, a nonce (r, Y) is jointly generated for next signature generation. The group element Y has a dual role, it is both Srv's contribution to the ephemeral key and to the nonce in next signature generation.

3.2.3 SCM Signature Generation

A signature generation is as in Protocol 2.

A. The client Clt, with clear text m' does the following:
1) Lookup the record (u, r, Y, pk); $h_Y \leftarrow H_2(Y)$;
2) $x_1, x'_1, r' \leftarrow_R [p-1]$; $X_1 \leftarrow G^{x_1}$; $t \leftarrow H_2(X_1, Y)$;
3) $z \leftarrow tx_1 \bmod p$; $Z \leftarrow X_1^t$; $X \leftarrow ZY$; $R \leftarrow H_1(X)$;
4) $V_1 \leftarrow R^z$; $X'_1 \leftarrow G^{x'_1}$; $R_1 \leftarrow H_1(X'_1, \text{``}S\text{''})$; $V'_1 \leftarrow R_1^{x'_1}$;
5) Get the password Pwd from the user;
6) $a_1 \leftarrow H_2(u, Pwd)$; $W_1 \leftarrow R^{a_1}$; $W'_1 \leftarrow R_1^{a_1}$;
7) $m \leftarrow H_2(m')$; $m_1 \leftarrow (r, r', m, R, V_1, W_1, X_1)$;
8) $h_1 \leftarrow H_2(m_1, G, X'_1, R_1, V'_1, W'_1, pk)$;
9) $\sigma_1 \leftarrow x'_1 + h_1 a_1 \bmod p$;
10) Send $(r, r', h_Y, m, X_1, V_1, W_1, W'_1, \sigma_1, h_1, pk)$ to Srv;

B. At receipt of $(r, r', h_Y, m, X_1 \neq 1_G, V_1, W_1, W'_1, \sigma_1, h_1, pk)$, Srv does the following:
1) Lookup the the record $(a_2, \hat{r}, \hat{y}, \hat{Y}, pk, T)$ with key pk;
2) $X'_1 \leftarrow G^{\sigma_1 + h_1 a_2} pk^{-h_1}$; $R_1 \leftarrow H_1(X'_1, \text{``}S\text{''})$;
3) $V'_1 \leftarrow R_1^{\sigma_1} {W'_1}^{-h_1}$; $t \leftarrow H_2(X_1, \hat{Y})$; $X \leftarrow X_1^t \hat{Y}$; $R \leftarrow H_1(X)$;
4) If $h_1 = H_2((r, r', m, R, V_1, W_1, X_1), G, X'_1, R_1, V'_1, W'_1, pk)$ then
   a) If $\hat{r} = r$ and $h_Y = H_2(\hat{Y})$ then
      i. $V_2 \leftarrow R^{\hat{y}}$; $W_2 \leftarrow R^{a_2}$; $V \leftarrow V_1 V_2$; $W \leftarrow W_1 W_2$;
      ii. $h \leftarrow H_2(m, G, X, R, V, W, pk)$; $\sigma_2 \leftarrow \hat{y} + h a_2 \bmod p$;
      iii. $y' \leftarrow_R [p-1]$; $Y' \leftarrow G^{y'}$; $T_S \leftarrow T_0$;
      iv. Store $(a_2, r', y', Y', pk, T_S)$ and send $(m, Y, Y', V_2, W_2, \sigma_2)$ to Clt.
   b) Else, deactivate Clt. *It is likely that a clone of Clt's private share exists.*
5) Else *$h_1 \neq h'_1$, the signature validation fails*
   a) If $\hat{r} = r$ and $h_Y = H_2(\hat{Y})$ then
      i. $y' \leftarrow_R [p-1]$; $Y' \leftarrow G^{y'}$; $T_S \leftarrow T_S - 1$;
      ii. Store $(a_2, r', y', Y', pk, T_S)$; and send $(0, Y, Y')$ to Clt;
      iii. If $T_S = 0$ then deactivate Clt.
   b) Else, deactivate Clt. *Srv detects a clone of Clt's password camouflaged private share.*

C. At receipt of a message msg from Srv, Clt does the following:
1) If msg parses as $(m, \hat{Y} = Y, Y', V_2, W_2, \sigma_2) \in [p-1] \times \mathcal{G}^4 \times [p-1]$ then
   a) $V \leftarrow V_1 V_2$; $W \leftarrow W_1 W_2$; $h \leftarrow H_2(m, G, X, R, V, W, pk)$;
   b) $\sigma_3 \leftarrow z + h \cdot a_1 \bmod p$; $\sigma \leftarrow \sigma_2 + \sigma_3 \bmod p$.
   c) Store (u, r', Y', pk) and delete the other values.
   d) Outputs (W, σ, h) as a signature on m'. *Clt can verify the validity of Srv's contribution by checking that $Vrfy_{CM}(pk, H_2(m'), (W, \sigma, h)) = 1$.*
2) Else if msg parses as $(0, Y, Y')$ then store (u, r', Y', pk) and delete the other values. *Probably an erroneous password was typed.*

D. Else, store (u, r', Y, pk) and delete the other values.

Protocol 2 SCM Signature Generation

The Protocol 2 contains three stages. In the first stage, step A, Clt generates X_1, which is its contribution to the ephemeral key, together with its contribution to the next nonce r'. Then, it computes $t \leftarrow H_2(X_1, Y)$, wherein Y is Srv's contribution to the ephemeral key, $z \leftarrow x_1 t$, $Z \leftarrow X_1^t$, $X \leftarrow ZY$ and $R = H_1(X)$. From this, it derives $V_1 = CDH(R, Z)$, $W_1 = CDH(R, A_1)$, and $(W'_1, \sigma_1, h_1) \leftarrow Sign_{VCM(\text{``}S\text{''}, A_2)}(a_1, (r, r', H_2(m'), R, V_1, W_1, X_1))$.

Notice that Clt does not send Y, it sends $H_2(Y)$. Doing so, the party communicating with Clt on behalf of Srv has to show that it knows Y. Hence, not only Srv may detect clones of Clt, but Clt may also detect attempts to impersonate Srv to it.

In step B, Srv computes $R \leftarrow H_1(X)$ and $Vrfy_{VCM(\text{``}S\text{''}, A_2)}(A_1, (r, r', m, R, V_1, W_1, X_1), (W'_1, \sigma_1, h_1))$. As $\sigma_1 = x'_1 + h_1 a_1$, and

$$G^{\sigma_1 + h_1 a_2} pk^{-h_1} = G^{\sigma_1} (G^{-a_2} pk)^{-h_1} = G^{\sigma_1} A_1^{-h_1},$$

at the steps B.2–B.4, Srv validates Clt's signature with regard to A_1. Then, Srv generates and sends its contribution to the shared signature.

In the last stage, using Srv's response, Clt completes the signature generation. As $sk = a_1 + a_2 \bmod p$ and $X = Z\hat{Y}$, it follows that $x = z + \hat{y} \bmod p$, $V = R^z R^{\hat{y}} = R^x$, $W = W_1 W_2 = R^{a_1} R^{a_2} = R^{sk}$, and $\sigma = (a + h a_1) + (\hat{y} + h a_2) = x + h \cdot sk$, where $h = H_2(m, G, X, R, V, W, pk)$, and (W, σ, h) is a SCM signature on m'.

3.2.4 SCM Signature Verification

To validate a SCM signature (W, σ, h) on m' (a CM signature on $H_2(m')$) with regard to pk, it suffices to execute $Vrfy_{CM}(pk, H_2(m'), (W, \sigma, h))$.

Remark 2 *If the message m' needs to be hidden to the server, m can be computed as $m \leftarrow H_2(m', t)$, for some random t chosen from a sufficiently large set.*

3.2.5 SCM Password Update

A password update can be performed as in Protocol 3. In Protocol 3, from his new password Pwd' and a fresh token u', Clt generates a new private share $a'_1 \leftarrow H_1(u', Pwd')$. It computes $Sign_{VCM(\text{``}U\text{''}, A_2)}(a_1, (a'_1 - a_1 \bmod p, r, r'))$, wherein r' is its contribution to the next nonce. When Srv receives and validates Clt's signature, it updates its private share to $a'_2 \leftarrow a_2 - \delta \bmod p$, and sends to Clt its contribution to the next nonce. After a successful update, it still holds that $a'_1 + a'_2 = sk \bmod p$.

A. The client Clt does the following:
   1) Lookup the record (u, r, Y, pk);
   2) $x'_1, r', u' \leftarrow_R [p-1]$; $X'_1 \leftarrow G^{x'_1}$; $h_Y \leftarrow H_2(Y)$;
   3) Get the password Pwd and the new password Pwd' from the user;
   4) $a_1 \leftarrow H_2(u, Pwd)$; $a'_1 \leftarrow H_2(u', Pwd')$; $\delta \leftarrow a'_1 - a_1 \bmod p$;
   5) $R_1 \leftarrow H_1(X'_1, ``U")$; $V'_1 \leftarrow R_1^{x'_1}$; $W'_1 \leftarrow R_1^{a_1}$;
   6) $h_1 \leftarrow H_2((\delta, r, r'), G, X'_1, R_1, V'_1, W'_1, pk)$; $\sigma_1 \leftarrow x'_1 + h_1 a_1 \bmod p$;
   7) Send $(\delta, r, r', h_Y, W'_1, \sigma_1, h_1, pk)$ to the server Srv;

B. At receipt of $(\delta, r, r', h_Y, W'_1, \sigma_1, h_1, pk)$, Srv does the following:
   1) Lookup the the record $(a_2, \hat{r}, \hat{y}, \hat{Y}, pk, T)$ with key pk;
   2) If $r = \hat{r}$ and $h_Y = H_2(\hat{Y})$ then
      a) $X'_1 \leftarrow G^{\sigma_1 + h_1 a_2} pk^{-h_1}$; $R_1 \leftarrow H_1(X'_1, ``U")$;
      b) $V'_1 \leftarrow R_1^{\sigma_1} {W'_1}^{-h_1}$; $y' \leftarrow_R [p-1]$; $Y' \leftarrow G^{y'}$;
      c) If $h_1 = H_2((\delta, r, r'), G, X'_1, R_1, V'_1, W'_1, pk)$ then $T_S \leftarrow T_0$; store $(a_2 - \delta \bmod p, r', y', pk, T_S)$ and send $(1, Y, Y')$ to Clt.
      d) Else
         i. $T_S \leftarrow T_S - 1$; store (a_2, r', y', pk, T_S) and send $(0, Y, Y')$ to Clt;
         ii. If $T_S = 0$ then deactivate Clt.
   3) Else, deactivate Clt.

C. At receipt of msg from the server, Clt does the following:
   1) If msg parses as $(1, Y, Y') \in \mathcal{G}^2$ then store (u', r', Y', pk);
   2) Else store (u, r', Y', pk). *The update failed.*

Protocol 3 SCM Password Udate

4 Security and Efficiency Comparisons

4.1 Security Comparisons

As far as we are aware, the work that seems closer to ours in its aims and achievements is probably the Smart–ID design [4]. Smart–ID offers no password update protocol, and as shown in [9], its clone detection mechanism is flawed. So, we consider Smart–ID with the clone detection mechanism from [9]. Recall that the aim of SCM is a secure server supported signature scheme, which (*i*) is geared to password based software only implementations in server supported settings, (*ii*) integrates a secure clone detection mechanism, (*iii*) offers a shared key pair generation to avoid the existence of a reference point for offline dictionary attacks, and (*iv*) offers a password update protocol. For any $m' \in \mathbf{M}$ and $(sk, pk) \in \{Gen(dp)\}$, $Sign_{SCM}(sk, m')$ is defined as $Sign_{CM}(sk, H_2(m'))$, SCM's security in the $UF - CMA$ model is essentially that of CM. Theorem 1 follows straightforwardly from [5, Theorem 2].

Theorem 1 *We assume the RO model. Let* $q_{H_i, i=1,2}$ *and* q_s *be respectively the number of times* $\mathcal{A}$ *issues the* $H_{i,i=1,2}$ *and* $Sign$ *oracles in Game . If the* CDH *problem is* $(\tau(k), \varepsilon_{CDH}(k))$*–hard in* $\mathcal{G}$*, then* SCM *is* (τ, q_s, ε) *unforgeable against chosen message attacks, where*

$$\varepsilon \leqslant \varepsilon_{CDH} + q_{H_2}^2/2p + 2q_s(q_s + q_{H_1})/p.$$

Under the Random Oracle model and the CDH assumption SCM is (*i*) secure against a malicious client; (*ii*) it is secure against a malicious server, and (*iii*) is also secure against device read. We defer the detailed security proofs to the extended version of this paper.

We summarize in Table 1 the security comparison between SCM, Smart–ID, and a smart–card. We use the following short–hands (*a*) IGKS stands for Independent Generation of the Key Shares, (*b*) CD for Clone Detection, (*c*) PU for Password Update, (*d*) MS for Malicious Server, (*e*) DR for Device Read, (*f*) DMR for Device Memory Read, (*g*) DMRKG for Device Memory Read during Key Generation, (*h*) DM for Device Malware, (*i*) S for Secure, (*j*) V for Vulnerable, and (*k*) LV for Limited Vulnerability.

Table 1 Security comparison between SCM, Smart-ID, and a smart-card

	IGKS	CD	PU	MS	DR	DMR	DMRKG	DM
Smart-Card	no	no	yes	S	S	S	S	V
Smart-ID	yes	yes	no	S	S	LV	LV	V
SCM	yes	yes	yes	S	S	LV	LV	V

All the three solutions, are vulnerable to a resident malware. Our design achieves all the security goals realized in Smart–ID, in addition to providing a secure password update protocol.

4.2 Efficiency Comparisons

We neglect group membership tests, as they have a negligible cost in $\mathbb{Z}_q^*$ and elliptic curve groups. A SCM key pair generation requires two exponentiations, one at Clt and one at Srv. An execution of the signature generation protocol requires $Exp(\mathcal{G}, 6)$ operations at Clt, and two multi–exponentiations and $Exp(\mathcal{G}, 4)$ operations at Srv. Assuming that a multi–exponentiation cost is about $Exp(\mathcal{G}, 1.25)$ [8, Algorithm 14.88], the computational effort of the server is about $Exp(\mathcal{G}, 6.5)$ operations. A signature verification requires two multi–exponentiations, about $Exp(\mathcal{G}, 2.5)$ operations. A SCM signature consists in one group element and two $|p|$–bits integers.

A Smart–ID key pair generation requires the generation of two RSA moduli, n_1 and n_2 at the client and server, respectively. A signature generation requires $Exp(\mathbb{Z}_{n_1})$ operations at Clt and $Exp(\mathbb{Z}_{n_1}) + Exp(\mathbb{Z}_{n_2}) \approx Exp(\mathbb{Z}_{n_2}, 2)$ operations at Srv. A signature verification requires one exponentiation in $\mathbb{Z}_{n_1 n_2}$, however the exponent can be chosen to be sparse ($2^{16} + 1$, for instance), and the exponentiation negligible. A Smart–ID signature is a $2|n_1|$–bits integer.

Given that both SCM and Smart–ID's security reductions are tight, for a target security of 128–bits, for instance, the RSA moduli n_1 and n_2 need to have a bitlength ≈ 3072 [10]. In contrast, SCM can be instantiated over an elliptic curve (sub)group $\mathcal{G} = E(\mathbb{F}_q)$ such that $|q| \approx 256$ [10]. Considering a

classical square–and–multiply based exponentiation, $Exp(\mathbb{Z}_{n_1}) \approx 4072 \cdot Mult(\mathbb{Z}_{n_1})$, where $Mult(\mathbb{Z}_{n_1})$ denotes the cost of a multiplication in $\mathbb{Z}_{n_1}$. Assuming that a group operation in $\mathcal{G}$ requires $13 \cdot Mult(\mathbb{F}_q)$ [7, p. 96], an exponentiation in $\mathcal{G}$ requires $Exp(\mathcal{G}) \approx 256 \times 1.5 \times 13 \approx 4992 \cdot Mult(\mathbb{F}_q)$. Using Karatsuba–Ofman's algorithm [3, chap 1], $Mult(\mathbb{Z}_{n_1}) \approx |n_1|^{1.585} \approx 51 \times |q|^{1.585} \approx 51 \cdot Mult(\mathbb{F}_q)$, and then $Exp(\mathbb{Z}_{n_1}) \approx 40 \cdot Exp(\mathcal{G})$.

For a target of 128–bits of security, a SCM signature generation over ECC is expected to be 6.6 times faster than Smart–ID at client side, and 12.3 times faster at server side, and these values are positively correlated to the number of bits of security. For sparse public exponents, signature verification is faster in Smart–ID than in SCM. A Smart–ID signature has 6144–bits length, while a SCM signature, requires only 1024 bits (indeed 768 in a compact representation the ECC group elements). SCM signatures are 6 times smaller than the Smart–ID ones.

5 Concluding Remarks

We proposed a password based server supported signature scheme, built from the Chevallier Mames' scheme, termed SCM, geared to mobile devices. Under the RO model and the CDH assumption, SCM is (*i*) unforgeable against chosen message attacks, (*ii*) secure against a malicious client, (*iii*) secure against a malicious server, and (*iv*) secure against device read, in addition to offering a secure clone detection mechanism, and a secure password update protocol.

A SCM signature generation requires 6 exponentiations at the client and 4 exponentiations at the server. For a target of 128–bits of security, we expect SCM to be 6 times faster than Smart–ID, at client side, and 12.3 times faster at server side. And, SCM signatures are about 6 times smaller than the Smart–ID ones.

In a forthcoming stage, we will be interested in password based server supported signcryption schemes, geared to mobile devices.

References

[1] Bellare, M., Rogaway, P.: Random oracle are practical: A paradigm for designing efficient protocols. In: ACM–CCS 1993, pp. 62–73. ACM (1993)

[2] Boyd C., Mathuria A., Stebila D.: Password-Based Protocols. In: Protocols for Authentication and Key Establishment. Information Security and Cryptography. Springer, Berlin, Heidelberg (2020)

[3] Brent, R. P., Zimmermann, P.: Modern computer arithmetic (Vol. 18). Cambridge University Press (2010)

[4] Buldas A., Kalu A., Laud P., Oruaas M.: Server-Supported RSA Signatures for Mobile Devices. In: Foley S., Gollmann D., Snekkenes E. (eds) Computer Security — ESORICS 2017. ESORICS 2017. LNCS, vol 10492. Springer, Cham (2017)

[5] Chevallier–Mames B.: An Efficient CDH–Based Signature Scheme with a Tight Security Reduction. In: Shoup V. (eds) Advances in Cryptology — CRYPTO 2005. CRYPTO 2005. LNCS, vol 3621. Springer, Berlin, Heidelberg (2005)

[6] Damgård, I., Mikkelsen, G. L., Skeltved, T.: On the security of distributed multi–prime RSA. In: Lee J., Kim J. (eds) Information Security and Cryptology — ICISC 2014. ICISC 2014. LNCS, vol 8949. Springer, Cham (2015)

[7] Hankerson D., Menezes A. J., Vanstone S.: Guide to elliptic curve cryptography. Springer (2004)

[8] Menezes A., van Oorschot P., Vanstone S.: Handbook of Applied Cryptography. CRC Press (1996)

[9] Sarr, A. P.: Cryptanalysis and Improvement of Smart–ID's Clone Detection Mechanism. Cryptology ePrint Archive: Report 2019/1412 (2019)

[10] Smart, N. P.: Algorithms, Key Size and Protocols Report (2018). ECRYPT–CSA, H2020-ICT-2014–Project, 645421 (2018)