Identification Systems
Lecture 3

Piotr Syga

2021-03-16

PACE – idea

J. Bender, M. Fischlin, D. Kügler, *Security Analysis of the PACE Key-Agreement Protocol*

Password Authenticated Connection Establishment

Let us assume that a chip (A) and a reader (B) share a password π chosen at random from a set of cardinality N. We want to generate the secret key K_π as an ephemeral key allowing message exchange using a *controlled* channel in order to establish permanent (session) keys.

PACE Protocol

A		B
password π		password π
authenticated EC parameters $\mathcal{G} = (a, b, p, q, G, k)$		
$K_\pi = \mathcal{H}(\pi \Vert 0)$ choose $s \leftarrow \mathbb{Z}_q$ $z = \mathcal{C}(K_\pi, s)$		$K_\pi = \mathcal{H}(\pi \Vert 0)$
	$\xrightarrow{\mathcal{G}, z}$	abort if $\mathcal{G}$ incorrect $s = \mathcal{C}^{-1}(K_\pi, z)$
..... Map2Point(s)		
	$\longrightarrow$ $\longleftarrow$	
jointly generate $\widehat{G}$		
choose $y_A \leftarrow \mathbb{Z}_q^*$ $Y_A = y_A \cdot \widehat{G}$		choose $y_B \leftarrow \mathbb{Z}_q^*$ $Y_B = y_B \cdot \widehat{G}$
	$\xleftarrow{Y_B}$	
abort if $Y_B \notin \langle G \rangle \setminus \{0\}$		
	$\xrightarrow{Y_A}$	abort if $Y_A \notin \langle G \rangle \setminus \{0\}$
$K = y_A \cdot Y_B$ $K_{\text{enc}} = \mathcal{H}(K \Vert 1)$ $K_{\text{mac}} = \mathcal{H}(K \Vert 2)$ $K'_{\text{mac}} = \mathcal{H}(K \Vert 3)$ $T_A \leftarrow \mathcal{M}(K'_{\text{mac}}, (Y_B, \mathcal{G}))$		$K = y_B \cdot Y_A$ $K_{\text{enc}} = \mathcal{H}(K \Vert 1)$ $K_{\text{mac}} = \mathcal{H}(K \Vert 2)$ $K'_{\text{mac}} = \mathcal{H}(K \Vert 3)$ $T_B \leftarrow \mathcal{M}(K'_{\text{mac}}, (Y_A, \mathcal{G}))$
	$\xleftarrow{T_B}$	
abort if T_B invalid	$\xrightarrow{T_A}$	abort if T_A invalid
key$= (K_{\text{enc}}, K_{\text{mac}})$ sid $= (Y_A, Y_B, \mathcal{G})$ pid $= \epsilon$		key$= (K_{\text{enc}}, K_{\text{mac}})$ sid $= (Y_A, Y_B, \mathcal{G})$ pid $= \epsilon$

Map2Point

Several versions of Map2Point procedure has been proposed, to obtain a group generator $\hat{G}$. **Diffie-Hellman Mapping**
Both parties calculate $H = x_a x_b G$ with DH exchange of $X_a = x_a G$ and $X_b = x_b G$ and derrive $\hat{G} = sG + H$ with checking for the trivial cases.

DH2Point(s)

A		B
choose $x_A \leftarrow \mathbb{Z}_q^*$		choose $x_B \leftarrow \mathbb{Z}_q^*$
$X_A = x_A \cdot G$		$X_B = x_B \cdot G$
	$\xleftarrow{X_B}$	
	$\xrightarrow{X_A}$	
abort if $X_B = 0$		abort if $X_A = 0$
$H = x_A \cdot X_B$		$H = x_B \cdot X_A$
$\widehat{G} = sG + H$		$\widehat{G} = sG + H$

Map2Point

Several versions of Map2Point procedure has been proposed, to obtain a group generator $\hat{G}$.

Coin-flipping Mapping

A generates $X_a = x_a G$ and presents $H(X_a)$, B transmits similarly obtained X_b (in plain), and A transmits their value, which is checked against the commitment. $H = X_a + X_b$ and $\hat{G} = sG + H$.

Coin2Point(s)

A		B
choose $x_A \leftarrow \mathbb{Z}_q^*$		choose $x_B \leftarrow \mathbb{Z}_q^*$
$X_A = x_A \cdot G$		$X_B = x_B \cdot G$
$C_A = \mathcal{H}(X_A)$	$\xrightarrow{C_A}$	
	$\xleftarrow{X_B}$	
	$\xrightarrow{X_A}$	
		abort if $C_A \neq \mathcal{H}(X_A)$
$H = X_A + X_B$		$H = X_A + X_B$
$\widehat{G} = sG + H$		$\widehat{G} = sG + H$

Map2Point

Several versions of Map2Point procedure has been proposed, to obtain a group generator $\hat{G}$.

Hash-into-the-Curve Mapping

We assume efficient function *h2c* that allows projecting a string directly into the curve and PRNG R. B generates and sends another nonce t and generate a random string using $R(s, t)$.

Hash2Point(s)
(assumes h2c function onto curve
and pseudorandom function R)

A		B
		choose nonce t
	$\xleftarrow{t}$	
$\widehat{G} = \mathrm{h2c}(R(s,t))$		$\widehat{G} = \mathrm{h2c}(R(s,t))$

Map2Point

Several versions of Map2Point procedure has been proposed, to obtain a group generator $\hat{G}$.

Power-to-Group Mapping

A and B map s to element $s^w \mod p$, with $p = wq + 1$ and w being β-smooth for large β. H and $\hat{G}$ may be obtained using interactive generation or reencryption.

Power2Point(s) (with re-encryption)
(for $\mathbb{Z}_p^*$, $p = wq + 1$)

A		B
		choose key K'
		$s' = \mathcal{C}(K', s)$
		where $s = \mathcal{C}^{-1}(K_\pi, z)$
	$\xleftarrow{K'}$	
$s' = \mathcal{C}(K', s)$		
$\widehat{G} = (s')^w \mod p$		$\widehat{G} = (s')^w \mod p$

Security Analysis

Proof Sketch omitting FS and Corrupt queries.

Game 0 : Original attack on the protocol

Game 1 : We abort whenever there are $\pi \neq \pi'$ that yield the same K_π. As there are $\frac{N^2}{2}$ such pairs in ROM
$\Pr[Game0] \leq \Pr[Game1] + \frac{N^2}{2|\mathrm{Range}(\mathcal{H})|}$.

Game 2 : We abort whenever there are $K_\pi \neq K'_\pi$ such that $\mathcal{C}^{-1}(K_\pi, z) = \mathcal{C}^{-1}(K'_\pi, z)$, $\Pr[Game1] \leq \Pr[Game2] + \frac{q_e N^2}{2q}$

Game 3 : We abort whenever there are $K \neq K'$ of two accepting user instances that yield identical encryption or MAC keys.
$\Pr[Game2] \leq \Pr[Game3] + \frac{6q_e^2}{|\mathrm{Range}(\mathcal{H})|}$

Security Analysis

Proof cont.

Game 4 : We replace ideal cipher $\mathcal{C}$ with lazy-sampling of 2 lists. Let THQ be the event that the adversary makes a so-called target hash query about $K||n$ for some,where K is the key some honest user instance has derived. The bound on THQ is made using separate, gPACE-DH game.
$\Pr[Game3] \leq \Pr[THQ] + \Pr[Game4|\neg THQ] + \frac{q_c q_e}{q}$

Game 5 : We even declare the adversary victorious if it ever submits a query $K||n$ for $n \in \{1, 2, 3\}$ to the hash oracle $\mathcal{H}$ such that a user instance $(U; i)$ has computed this key and sent out the final MAC. This cannot happen with probability of a single trial higher than $\frac{q_e}{N}$ (guessing at which query the adversary chooses the correct trial and run honest Map2Point).
$\Pr[THQ] \leq \frac{q_e}{N} + q_e(\mathrm{Adv}^{\mathrm{gPACE-DH}}_{\mathrm{Map2Point}}(t', N, q_h))$

Security Analysis

Proof cont.

Game 6 : We replace K_{enc} and K_{mac} in queries by random keys. Note that, since we assume that the adversary never makes a hash query about a DH-key of an accepting user instance, this simulation is perfect unless there is an accepting instance $(U; i)$ having the same DH-key as another instance $(U'; j)$ and the instances are not partnered. If they are it breaches canonicality. Since we guess the first Execute query with probability $1/q_e$ and the find a valid forge, we prevent the user from submitting K to MAC Oracle. Hence, $\mathcal{A}$ can only inject values $(Y_A; G)$, $(Y_B; G)$,$(Y'_A; G')$, $(Y'_B; G')$ with valid MACs generated by honest users in other executions, but since we guessed and K is determined, y_* at random.
$\Pr[Game5|\neg THQ] \leq \Pr[Game6] + \frac{2q_e}{q} + q_e \mathrm{Adv}^{\mathrm{forge}}_{\mathcal{M}}(t', 2q_e)$

Final Game : Al keys returned in Test are random, hence the advantage is bounded by $\frac{1}{2}$.

Final notes

- Forward security requires modification of gPACE-DH problem
- PACE uphold untraceability in a restricted sense – issuers metadata like signature of authentication signautre or EC used allows partial tracking
- Unlinkability between two eavesdropped communication is provided
- Non-transferability follows from the fact that the terminal could generate the data in a communication itself with the help of the password

Further research

- Mostly focused on relaxing the security assumptions w.r.t. groups, e.g. showing version of the protocol that is secure even in groups with a DDH oracle (cf. groups with admissible bilinear maps)
- Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: The PACE–AA protocol for machine readable travel document, and its security. – using signatures to mitigate impersonation attacks, maintains AKE security
- Bender, J., Fischlin, M., Kügler, D.: The PACE|CA protocol for machine readable travel documents. and Hanzlik, L., Krzywiecki, Ł., Kutyłowski, M.: Simplified PACE|AA protocol.– binding executions with public keys by coin-knowledge — leads to ICAO SAC standard PACE-CAM
- Hanzlik, L., Kutyłowski, M.: Chip Authentication for E-Passports: PACE with Chip Authentication Mapping v2 – independence of mapping used

Definitions

Bilinear Mapping

Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, $(\mathbb{G}_T, \cdot)$ be cyclic groups of a prime order q. Let P, Q be generators of respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. We say that $(G1, G2, GT)$ are bilinear map groups if there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfying the following properties:

Bilinearity: $\forall_{(S,T)\in\mathbb{G}_1\times\mathbb{G}_2;\ a,\, b\in\mathbb{Z}_q} e(aS, bT) = e(S, T)^{ab}$

Non-degeneracy: $e(P, Q) \neq 1$ is a generator of $\mathbb{G}_T$

Computability: $\forall_{(S,T)\in\mathbb{G}_1\times\mathbb{G}_2;}$ $e(S, T)$ is efficiently computable

Definitions

Key Derivation Function

Key derivation functions are used to derive keys in a way that is indistinguishable from probing with the uniform distribution in the key space. The ICAO standard defines: $KDF(K, c) = H(K||c)$, where K is the shared secret, c is a counter and H is a hash function.
The following notation is used in the original paper:
$KDF_{Enc}(K) = KDF(K, 1)$, $KDF_{Mac}(K) = KDF(K, 2)$ and $KDF_{\pi}(\pi) = KDF(f(\pi), 3)$, where f is some encoding of password π defined in the standard.

Definitions

l-Strong Diffie-Hellman

Given $(l+3)$ elements $(g_1, g_1^x, \ldots, g_1^{x^l}, g_2, g_2^x) \in \mathbb{G}_1^{l+1} \times \mathbb{G}_2^2$, output $(c, g_1^{\frac{1}{(x+c)}}) \in \mathbb{Z}_q \times \mathbb{G}_1$. We say that an algorithm $\mathcal{A}$ has an advantage ε in solving the l-SDH in $\mathbb{G}_1, \mathbb{G}_2$ of prime orders q if:

$$\Pr[(c, g_1^{\frac{1}{(x+c)}}) \leftarrow \mathcal{A}(g_1, g_1^x, \ldots, g_1^{x^l}, g_2, g_2^x)] \geq \varepsilon \ ,$$

where the probability is taken over the random choice of the generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, the random choice of $x \in \mathbb{Z}_q$, and the random bits of $\mathcal{A}$. By $Adv^{l-SDH}(t)$ we denote the maximal advantage for any adversary running in time t in solving the l-Strong Diffie-Hellman problem.

The Oracles

The model implements the oracles similarly to the original paper:

Execute(C^i, S^j) – this query allows to model passive adversaries and outputs all between client and server.

Send(U^i, m) – this query allows to model an active adversary and outputs the message that U^i would generate upon receiving m.

Corrupt.pw(U) – returns the secret password π of U.

Corrupt.key(U) – returns the secret key sk_C of user $U \in \mathcal{C}$.

The Oracles

The model implements the oracles similarly to the original paper:

Reveal(U, i) – reveals the session key computed by U^i in an accepting state.

Register(C, pk) – allows to register a public key pk in the name of a new, adversarially controlled, client $C \in \mathcal{C}$.

Test(U, i) – initializes RoR game with a bit b, we assume that at some point of the game, the adversary makes a test query about (U, i) and that U^i terminates in accepting state with session key K. The oracle returns K if $b = 0$ or a random K' if $b = 1$, we can assume that the adversary never queries twice for the same user instance.

The protocol

ePassport:		Reader:
password π spar $= (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, pk_{CA})$ $sk_C^* = sk_C$ $pk_C^* = (pk_C, cert_C)$		password π spar $= (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, pk_{CA})$
$K_\pi = \mathsf{KDF}_\pi(\pi)$ choose $s \leftarrow_R \{0,1\}^\ell \subseteq \mathbb{Z}_q$ $z = \mathsf{Enc}(K_\pi, s)$	$\xrightarrow{z}$	$K_\pi = \mathsf{KDF}_\pi(\pi)$ $s = \mathsf{Dec}(K_\pi, z)$
Mapping Function ($\mathbf{Map_{GM}}$ or $\mathbf{Map_{IM}}$)		
create parameters $\hat{\mathbb{G}_1}$ as $(a_1,b_1,p_1,q_1,\hat{g_1},k_1)$=$\mathbf{Map}(\mathbb{G}_1, s)$		create parameters $\hat{\mathbb{G}_1}$ as $(a_1,b_1,p_1,q_1,\hat{g_1},k_1)$=$\mathbf{Map}(\mathbb{G}_1, s)$
Generic Mapping ($\mathbf{Map_{GM}}$)		
choose $y_C \leftarrow_R \mathbb{Z}_q^*$ $Y_C = g_1^{y_C}$	$\xleftarrow{Y_R}$	choose $y_R \leftarrow_R \mathbb{Z}_q^*$ $Y_R = g_1^{y_R}$
abort if $Y_R \notin \langle g_1 \rangle \backslash \{1\}$ $h = Y_R^{y_C}$ $\hat{g_1} = h \cdot g_1^s$	$\xrightarrow{Y_C}$	abort if $Y_C \notin \langle g_1 \rangle \backslash \{1\}$ $h = Y_C^{y_R}$ $\hat{g_1} = h \cdot g_1^s$
Integrated Mapping ($\mathbf{Map_{IM}}$)		
	$\xleftarrow{r}$	choose $r \leftarrow_R \mathbb{Z}_q^*$
$\hat{g_1} = \mathsf{H}_{EC}(s, r)$		$\hat{g_1} = \mathsf{H}_{EC}(s, r)$
choose $y'_C \leftarrow_R \mathbb{Z}_q^*$ $Y'_C = \hat{g_1}^{y'_C}$	$\xleftarrow{Y'_R}$	choose $y'_R \leftarrow_R \mathbb{Z}_q^*$ $Y'_R = \hat{g_1}^{y'_R}$
abort if $Y'_R \notin \langle g_1 \rangle \backslash \{1_{\mathbb{G}_1}\}$ $K = Y'^{y'_C}_R$ $K_{\mathsf{Enc}} = \mathsf{KDF}_{\mathsf{Enc}}(K)$ $K_{\mathsf{Mac}} = \mathsf{KDF}_{\mathsf{Mac}}(K)$ $K'_{\mathsf{Enc}} = \mathsf{KDF}(K, 2)$ $K'_{\mathsf{Mac}} = \mathsf{KDF}(K, 4)$ $T_C = \mathsf{Mac}(K'_{\mathsf{Mac}}, (Y'_R, \mathbb{G}_1))$	$\xrightarrow{Y'_C}$ $\xleftarrow{T_R}$	abort if $Y'_C \notin \langle g_1 \rangle \backslash \{1_{\mathbb{G}_1}\}$ $K = Y'^{y'_R}_C$ $K_{\mathsf{Enc}} = \mathsf{KDF}_{\mathsf{Enc}}(K)$ $K_{\mathsf{Mac}} = \mathsf{KDF}_{\mathsf{Mac}}(K)$ $K'_{\mathsf{Enc}} = \mathsf{KDF}(K, 2)$ $K'_{\mathsf{Mac}} = \mathsf{KDF}(K, 4)$ $T_R = \mathsf{Mac}(K'_{\mathsf{Mac}}, (Y'_C, \mathbb{G}_1))$
if $\mathsf{MVer}(K'_{\mathsf{Mac}}, (Y'_C, \mathbb{G}), T_R) = 0$, then abort $w = y'_C / sk_C$ $c = \mathsf{Enc}(K'_{\mathsf{Enc}}, (w, cert_C))$	$\xrightarrow{T_C}$ $\xrightarrow{c}$	if $\mathsf{MVer}(K'_{\mathsf{Mac}}, (Y'_R, \mathbb{G}), T_C) = 0$, then abort $(w, cert_C) = \mathsf{Dec}(K'_{\mathsf{Enc}}, c)$ abort if $\mathsf{CVer}(pk_{CA}, cert_C) = 0$ extract pk_C from $cert_C$ abort if $e(Y'_C, g_2) \neq e(\hat{g_1}, pk_C)^w$
key $= (K_{\mathsf{Enc}}, K_{\mathsf{Mac}})$ sid $= (Y'_C, Y'_R, \hat{\mathbb{G}_1}, \mathbb{G}_1)$ pid $= pk_C^*$		key $= (K_{\mathsf{Enc}}, K_{\mathsf{Mac}})$ sid $= (Y'_C, Y'_R, \hat{\mathbb{G}_1}, \mathbb{G}_1)$ pid $= pk_C^*$

Security

Theorem

In the random oracle model PACE-CAM2 satisfies:

$$Adv^{ike}_{PACE-CAM2}(t,Q) \leq q_e^2 q_h Adv^{2-SDH}(t') + \frac{q_h}{2^\lambda} + \frac{q_e^2}{q} + \\ +2q_e Adv^{forge}_{MAC}(t', 2q_e, 2q_e) + Adv^{forge}_{CA}(t', q_e)$$

Security game proof

Game0 : Original attack on the protocol

Game1 : We abort the game in case any honest reader accepts an unregistered public key. Since pk is stored in a certificate, we lower the advantage by the advantage of a successful forgery of a certificate.

Game2 : We abort if an honest ePassport computes the same key K that any honest ePassport has computed before. Since K depends on Y'_C (chosen at random by honest client) and Y'_R (chosen by possibly malicious reader) the probability that it matches any of the previous i keys, is at most $\frac{i}{q}$. It follows that the adversary's advantage can drop by a fraction of $\frac{q_e^2}{2q}$, where q_e is the number of executions.

Security game proof

Game3 : We abort if there are collisions among the values Y'_R computed by the honest readers. By the birthday bound this again lowers the adversary's advantage by $\frac{q_e^2}{2q}$.

Game4 : We abort if a malicious reader submits a valid T_R to the honest ePassport without hash query to the key K derived by the honest ePassport in the execution before and T_R was not computed by an honest reader. The adversary's loss can be at most q_e times the advantage of forging a MAC.

Game5 : We abort if a malicious ePassport submits a valid T_C to the honest reader without a hash query to the key K derived by the honest reader in the execution before and T_C was not computed by an honest ePassport. The adversary's loss can be at most q_e times the advantage of forging a MAC.